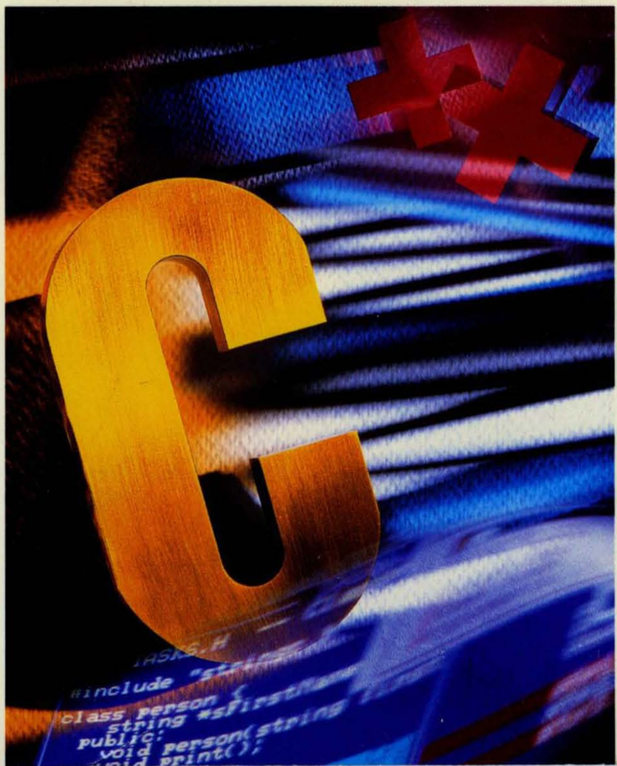
2.0

WHITEWATER
RESOURCE
TOOLKIT

BORLAND® C++



```
TASKS.H
#include "s
class person {
   string *sFirstNam
public:
   void person(string
   void print();
```

BORLAND

Borland® *C ++*
Version 2.0

Whitewater Resource Toolkit

BORLAND INTERNATIONAL, INC. 1800 GREEN HILLS ROAD
P.O. BOX 660001, SCOTTS VALLEY, CA 95067-0001

R1

PRINTED IN THE USA.
10 9 8 7 6 5 4 3 2

C O N T E N T S

T A B L E S

F I G U R E S

I N T R O D U C T I O N

Using the Resource Toolkit, you can see your resources as you edit them, instead of having to edit a script file that gets compiled later.

The Resource Toolkit provides the tools you need to build and modify resources used in applications designed to work under Microsoft's Windows 3.0. Those tools include editors for accelerators, bitmaps, cursors, icons, dialog boxes, menus, and strings; with them, you can create and edit resources. The Resource Toolkit also includes an organizing and efficiency tool called the Resource Manager. The Resource Manager is the central control for accessing the resource editors, and lets you accomplish other tasks besides, such as copying resources from one file into another, and so on.

With the Resource Toolkit editors, you can create and manage the user interfaces of your Windows applications while staying in Windows. This ability to work interactively saves you valuable time as you develop resources. Moreover, the Resource Toolkit can save resources in binary format. This eliminates the edit-compile-link cycle typical of resource development.

Features

The resources you can create, edit, and save with the Resource Toolkit are

- keyboard accelerators
- bitmaps
- cursors
- icons
- dialog boxes
- menus
- strings

In most languages, you can create these elements directly in the source code. However, by defining them as independent resources, you remove management responsibilities from the source code. Those management tasks are taken care of by Windows, resulting in more streamlined and efficient application code.

In addition to building resources, the Resource Toolkit lets you maintain C and C++ format header files and Pascal include files. Resources, header files, and include files are discussed in more detail in Chapter 2.

The main pieces of the toolkit

The Resource Toolkit can be logically divided into two major components: the main window (the Resource Manager), and the editors.

The Resource Manager

The main window in the Resource Toolkit is called the Resource Manager. This main window provides access to the toolkit's various specialized editors and to other functions you'll need. It is described in detail in Chapter 3.

Dialog Box editor

The Dialog Box editor is for creating and editing dialog boxes. When you create a new file in the Dialog Box editor, the editor automatically draws a captioned dialog box. In addition, the Dialog Box editor has a tool that you can use to create combo boxes, and supports the new owner-draw controls introduced in Windows 3.0. Combo boxes combine the features of an edit control and a list box; owner-draw controls are customized controls.

Bitmap, Cursor, and Icon editors

The Resource Toolkit supports the new bitmap formats Windows uses to create device-independent formats. The Bitmap, Cursor, and Icon editors read and write these new formats and fully support their color tables. In addition,

- The Bitmap editor lets you customize color palettes.

- The Icon and Cursor editors let you create multiple images of each icon or cursor resource. These multiple images make the icons and cursors device-independent. When the application makes a run-time call for an icon or cursor, Windows automatically loads the appropriate image for the device driver specified in its Setup program.
- You can use the Pouring tool, available in the Bitmap, Cursor, and Icon editors, to fill any area with color.
- The Bitmap, Cursor, and Icon editors have a Pixel Size tool which lets you draw with one of three pixel sizes: 1 by 1, 4 by 4, or 8 by 8.

Menu editor

The Menu editor supports the hierarchic menus introduced in Windows 3.0. Hierarchic menus let you create more than one pop-up menu level for each menu item. In the Menu editor, you create pop-up levels by indenting text in the Text field, and you define text for the pop-up items in the same table used to define text for the higher level items. This design makes it easier to create and edit pop-up menus. It also makes it easier to change any item's position on a menu, or to change its level within the menu hierarchy.

Other editors

The remaining editors are the Accelerator and String editors. The Accelerator editor lets you create and edit accelerator resources; accelerators are hot keys for issuing commands. The String editor lets you create and edit String resources, which are strings of text used by an application, usually for messages or captions in a window.

File formats you can work with

The files you can edit and save in the Resource Toolkit are discussed in more detail in Chapter 2.

You can use the Resource Toolkit for any application development project in Windows. It supports any Windows-compatible program (such as those written in Borland C++ 2.0) and can be used to maintain resources in executable (.EXE) files, resource (.RES) files, dynamic link libraries (DLLS), bitmap (.BMP) files, cursor (.CUR) files, and icon (.ICO) files.

Though the Resource Toolkit edits and saves files in binary format, it can also save resources in text files that are compatible with the resource script (.RC) files that the SDK uses to compile resources.

What you should know

The Resource Toolkit runs in Windows version 3.0 or later. To use the Resource Toolkit, you must know how to operate Windows. This manual assumes you're already familiar with Windows and a few Windows applications, and that you have ready access to the documentation that came with your Windows system. At a minimum, you should know how to

- use a mouse
- move and size windows, and make them active
- scroll through windows
- select options and files from menus, list boxes, and dialog boxes
- maximize a window, and restore the window to its previous size
- minimize a window or application to its icon
- use the Program Manager in Windows 3.0 or later

Hardware and software requirements

You must have Windows loaded. The Resource Toolkit runs on most computers that can run Windows. These include, but are not limited to, an IBM PC/AT, PS/2, or compatible, and 386-based computers such as the Compaq Deskpro.

To run the Resource Toolkit, your computer must have the following:

- Windows 3.0 or later.
- 1 MB RAM.
- Graphics display and adapter (EGA or better recommended).
- A mouse or other pointing device.
- A minimum of 700K disk space for temporary files. If you open large .EXE files, the temporary files created are as large as the .EXE file and require more disk space. For optimum performance, we recommend that you keep at least 3M free.

A printer is optional. All peripherals, including a mouse, graphics display and adapter, and printer, are supported through Windows. If your peripheral is supported by Windows, it will work with the Resource Toolkit. For example, the Resource Toolkit supports most of the IBM graphics adapters, as well as many others. Windows also supports a variety of mice and other pointing devices. The Windows Setup program displays a complete list of supported devices.

Installation

The Resource Toolkit is installed automatically by the Borland C++ installation program.

About this manual

This manual is organized into the following chapters:

Chapter 1, "Getting started," tells you how to begin and end a session in the Resource Toolkit.

Chapter 2, "About resources and files," provides an overview of resources, and discusses the files you can work with in the Resource Toolkit. It tells how to use those files to develop resources for a Windows application.

Chapter 3, "The Resource Manager," tells you how to use the Resource Toolkit's main window, the Resource Manager.

Chapters 4 through 9. Each of these chapters tells you how to use a specific Resource Toolkit editor.

Chapter 10, "Common keys and menus," tells you how to navigate and edit the tables in the Accelerator, String, and Menu editors. It also tells you how to use the menus whose options are common among all the editors: the File, Edit, and Header menus.

Appendix A, "Troubleshooting and error messages," contains information that may help you solve problems you might have while working with the Resource Toolkit; tells you how to recover disk space if you experience computer problems that unexpectedly end your editing session; and provides some information on controlling memory.

Where to now?

In order to get going in the Resource Toolkit, we recommend that you

- follow the instructions in Chapter 1 to learn how to start and exit from the Resource Toolkit
- proceed to Chapter 2 for an overview of resources and how to use the Resource Toolkit
- read Chapter 3 for information on how to use the Resource Toolkit's main window, the Resource Manager
- read the remaining chapters as you need them to use the editors.

C H A P T E R

1

Getting started

This chapter tells you how to start and exit the Resource Toolkit. It also tells you how to add the Resource Toolkit's icon to the Windows Program Manager.

You can start a Resource Toolkit session—or multiple Resource Toolkit sessions—from the DOS prompt or from within Windows. Once you start the session, the Resource Manager is opened, as well as the copyright box. To close the copyright box, click the OK button or press *Enter*.

Starting from DOS

To start the Resource Toolkit from the DOS prompt,

1. Change the directory to the Resource Toolkit's main directory. If you called it \WRT, type

   ```
   CD \WRT
   ```

   and press *Enter*.
2. Once you're in the Resource Toolkit's main directory, type

   ```
   WIN WRT
   ```

   and press *Enter*. This starts a Resource Toolkit session. The files WRT.IMA, WRT.DAT, and WRT.FON must be in the same directory as the file WRT.EXE.

Starting within Windows

Within Windows, you can start the Resource Toolkit from the Program Manager, the File Manager, or the MS-DOS executive.

Starting from the Program Manager

To start the Resource Toolkit from the Program Manager,

1. Start Windows and make the Program Manager active.
2. Activate the window that contains the Resource Toolkit's icon. If you haven't added the Resource Toolkit icon to the Program Manager, see below for how to add the icon, and for how to exit without the icon.
3. Double-click the Resource Toolkit's icon.

Starting from the File Manager or MS-DOS executive

To start the Resource Toolkit from the File Manager or the MS-DOS executive,

1. Open the directory that contains the file WRT.EXE.
2. Double-click the file name. This starts a Resource Toolkit session. The files WRT.IMA, WRT.DAT, and WRT.FON must be in the same directory as file the WRT.EXE.

Adding the icon to the Program Manager

If you add the Resource Toolkit icon to the Program Manager, you can start the Resource Toolkit by double-clicking its icon.

To add the icon to the Program Manager,

1. Start Windows and make the Program Manager active.
2. Activate the window where you want to add the icon.
3. Choose File | New from the Program Manager's menu. This opens the New Program Object dialog box.
4. Select Program Item in the New field, then click the dialog box's OK button to accept the setting. This opens the Program Item Properties dialog box.
5. In the dialog box's Description field, type the name you want displayed under the icon. For example, you might want Resource Toolkit displayed under the icon.
6. In the dialog box's Command Line field, type

   [*path*] WRT.EXE

where the optional *path* is the directory that contains the file. For example,

```
C:\WRT\WRT.EXE
```

defines the path C:\WRT\ for WRT.EXE.

7. Click on the Change Icon button to change the icon, if desired.
8. Click the dialog box's OK button. This adds the icon to the active window.
9. To start the Resource Toolkit, double-click its icon.

Exiting from the Resource Toolkit

To exit from the Resource Toolkit,

1. Double-click the Control menu in the Resource Manager.
2. Alternatively, click the Control menu and choose Exit WRT from its menu.

C H A P T E R

2

About resources and files

See Chapters 4 through 9 for specific information on each editor.

This chapter provides an overview of resources and the files you can work with in the Resource Toolkit. It also tells how to use the files to develop resources for a Windows application.

What is a resource?

You can think of a resource as a separate data file that's appended to your executable file. As long as the resource is correctly defined, specific information about it can be changed without affecting the application.

Windows applications are composed of a number of different graphical elements. For example, a single application might have ten different mouse cursors representing different operational modes. Instead of having to explicitly store the cursor information in the application code, you can store it separately as a *resource* that can be loaded into the program when needed. The advantages of separating resources in this manner include:

- it makes it easier to use the same resource definitions for multiple applications
- the application's visual appearance can be modified without changing or even accessing the application's source code

Resources are data categorized by type—bitmap, cursor, dialog box, and so on—and assigned a unique name or number within the type. The data can be any information relevant to the resource. For example, a dialog box resource might contain data representing three buttons, the button labels, and where the buttons are to be drawn on the screen.

Resources describe the visual characteristics of program elements. They do not define functionality; in fact, functional code is unaffected by changes to the resources. When resources are separated from source code, the two can be manipulated independently. This means you can use the Resource Toolkit to change the look and feel of your Windows applications without accessing the source code.

You can create or edit most types of resources used by Windows programs; see page 15.

The Resource Toolkit lets you browse, edit, view, copy, and delete certain resources. The resources you can copy, rename, and delete but *not* edit or browse are

- **Fonts.** Fonts the application uses to display text.
- **.RC data files.** Raw data resources.
- **User-defined resources.** Any data that needs to be added to the executable file.

The next section covers the resources you can edit, view, copy, and delete.

Editable resources

This section describes the resource types you can edit in the Resource Toolkit. They are:

accelerators	dialog boxes
bitmaps	menus
cursors	strings
icons	

Accelerators

Accelerators are also commonly called shortcuts or hot keys.

An accelerator is a key or key combination you can press as an alternative to choosing a menu item to invoke a command. Accelerators bring speed and convenience to an application. For example, it's quicker to press *Shift-Ins* than it is to pull down the Edit menu and choose Paste.

The Accelerator editor lets you define accelerators by pressing the key or keys you want to substitute for each defined menu option. The accelerator can then be identified on a menu as an alternative means to invoke the menu operation.

Bitmaps

A bitmap is a body of data an application uses to display a picture on the screen. A bitmap can be of any dimension, and can consist of more than one color. A bitmap can fill an entire window (as in a Windows desktop image), or cover only a small area (as in a cursor).

The Bitmap editor is a sophisticated editing tool that brings the power of a typical paint program to the task of editing color bitmaps. It provides various line and shape drawing tools, as well as area fill, selection, and movement functions.

Cursors

A cursor is the marker used to select graphical elements on a screen, or to locate the insertion position for input. A cursor resource is a special type of bitmap that's confined to the specific dimension of 32 by 32 pixels. Because a cursor is used to indicate a specific point on the screen, it has the ability to define the unique pixel that maps to an exact screen location. This pixel is referred to as the *hot spot*. Cursors of varying shapes can be defined for an application.

Cursor resources contain the data for two bitmaps: one describing the color of the cursor at each point, and another describing how the cursor should appear against an existing background. This allows cursors to exhibit some interesting properties. For example, parts of a cursor can be made transparent onscreen, or the cursor's background color can change.

The Cursor editor is a specialized version of the Bitmap editor. It provides the same sophisticated tools as the Bitmap editor, but the tools are tailored to the specific needs of cursor editing.

Icons

An icon is a graphical screen element you can select to change an application's program state. For example, to start the Resource Toolkit, you click its icon in the Windows Program Manager.

An icon resource is another type of specialized bitmap. Like a cursor, an icon can define various combinations of foreground with background colors, so its appearance can change as it's moved across the screen.

The Icon editor is another specialized version of the Bitmap editor, this time tailored to the unique demands of icon editing.

Dialog boxes

A dialog box is an input screen that lets you choose among varied options in a program. To make selections, you manipulate graphical elements, called *controls,* in the dialog box. Controls are the familiar gadgets that make windowing applications so easy to use: list boxes, scroll bars, and entry fields, for example.

A dialog box resource is a complex set of data describing the properties of the dialog box and its controls. Among those properties are the resource's size, screen location, and text labels.

The Dialog Box editor lets you construct dialog boxes with the mouse. Thus, you see the dialog box exactly as it will appear in an application. The editor includes tools for drawing the box, as well as for inserting, sizing, and moving the dialog box controls.

Menus

A menu lists the available program options you can activate. When you choose an option from the list, either an action is taken, or another menu opens to provide additional options. A menu resource identifies the available options, and determines the order in which the options are listed.

The Menu editor is a sophisticated editor for defining menu resources. It lets you visually construct new pull-down menus, customize their appearance, and interactively test them during the editing session.

Strings

Strings are the text an application displays in its menus, dialog boxes, error messages, and so on. You can define string text within an application's source code, or as a separate resource that's loaded as needed into the application.

If you define strings as resources, you can make your application more flexible and save yourself work in a number of ways. For example,

- Defining strings as resources makes it easier to customize an application for use in another country. This is possible because the Resource Toolkit can then be used to translate the strings from one language into another (from English, say, into French) without requiring you to distribute the source code.

- Defining strings as resources also makes it easier to improve and edit an application's messages. For example, if an application has confusing or ambiguous messages, those messages could be clarified by editing their corresponding string text. This could be done without altering any of the application's source code.

The String editor lets you easily create and edit string resources.

Files you can edit or save

You can use the Resource Toolkit to create or edit most types of resource files used by Windows programs. In addition, because executable files may contain resources as well as source code, you can also use the Resource Toolkit to edit the resources of executable files, even if you don't have the source code. This section briefly discusses the types of files you can edit or save in the Resource Toolkit. For more information on any of these file types, see Charles Petzold's *Programming Windows.*

Table 2.1 shows the types of files you can edit in the Resource Toolkit.

Table 2.1
Files that can be edited in the Resource Toolkit

File	Type	Description
.EXE	executable	Executable file containing application resources and compiled program code
.RES	resource	Compiled resource file
.DLL	executable	Executable module containing application resources and compiled program code
.H	source code	Header file containing symbolic names for defined resources
.ICO	resource	Icon resource file
.CUR	resource	Cursor resource file
.BMP	resource	Bitmap resource file

.ICO, .CUR, and .BMP files can each contain only one resource

Table 2.2 shows the types of files that can be generated by the Resource Toolkit, but that have to be compiled by Microsoft's Resource Compiler to create .RES or .EXE files for editing.

Table 2.2
Files that can be generated by the Resource Toolkit

File	Type	Description
.DLG	resource script	Dialog box resource script file. A .DLG file generated by the Resource Toolkit can contain only a single dialog box resource definition in text format.
.RC	resource script	The resource script file. This file is created with an ASCII editor; it contains a list of all the application's defined resources.

Certain types of resource files can be copied, renamed, and deleted but *cannot* be created, browsed, or edited in the Resource Toolkit:

File	Type	Description
.FON	resource	Fonts the application uses to display text.
.DAT	resource	Raw data resources.

The Resource Toolkit can save resources directly into .RES and/or .EXE files. This means you don't need Microsoft's Resource Compiler to compile resources, nor do you need the compiler to add compiled resources to an .EXE file.

With the Resource Toolkit, developing resources for Windows applications is straightforward:

1. Create the application's first resource and save it in a new .RES file.
2. Create all the additional resources the application needs and save them into the same .RES file.
3. Once all resources are defined in the .RES file, copy them into the application's .EXE file.

Note Since the Resource Toolkit can write the resources directly into an .EXE file, you might wonder why you can't just eliminate the .RES file. The reason is that linking an .EXE file to your Windows and C/C++ run-time libraries clears all resources from the .EXE file. (Turbo Pascal automatically links in the resources at the link phase; the resources must be in a .RES file.) If the resources are stored separately in a .RES file, you can always add them back to an .EXE file after a re-link.

This process, depicted in Figure 2.1, eliminates several steps you would otherwise have to perform to create resources for your application. This is because you don't have to create .RC and .DLG script files, and you don't need the .RC compiler.

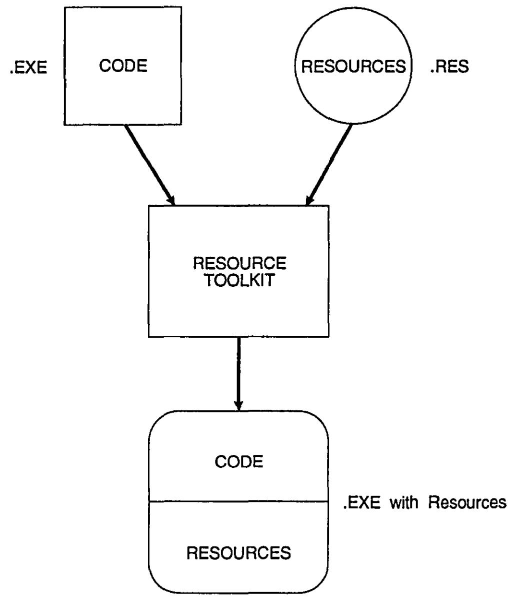


Figure 2.1 Building Windows applications in the Resource Toolkit

Executable files

An executable (.EXE) file contains all an application's resources plus compiled program code. This means .EXE files store all resource types. You can directly edit the resources in an .EXE file and save modified resources back into the file. This is useful for making cosmetic changes to applications, even if you don't have the source code. For example, you can change an application's icons, or divide a complex pop-up menu in the application into two or more simpler hierarchic menus.

Resource files

Resource (.RES) files contain any type or combination of resources in binary format. You can directly edit .RES files in the Resource Toolkit and save modified resources back into the file. You can also save a resource into a new .RES file. Typically, you would create one .RES file for each application, and store all the application's resources in that file.

Dynamic link libraries

A dynamic link library (.DLL) is an executable module containing resources and functions that can be called by Windows applications. A DLL is similar to a run-time library, except that it's linked to a program at run time rather than when conventional application files are linked. The library is activated when another module calls one of its functions. Combining commonly used routines in a DLL allows them to be accessed by multiple applications and programs; they do not have to be duplicated in every program that uses them.

You can modify routines in a DLL without relinking the DLL to the programs that call it. This contrasts with normal object libraries, which have to be relinked whenever their routines are modified.

Resource and dialog box script files

A resource script (.RC) file contains all an application's resources in text format. A dialog box resource script (.DLG) file contains only dialog box resources in text format. Both these types of files can be generated by the Resource Toolkit or created with a text editor.

.RC files identify resources by name or number, and define their attributes, styles, and other information relevant to the individual resources. Here's what part of the contents of an .RC file that was generated by the Menu editor might look like:

```
CWMenus MENU
BEGIN
  POPUP "&File"
    BEGIN
     MENUITEM "&New...\t^N",   CW_FILE_NEW
     MENUITEM "&Open...\t^O",  CW_FILE_OPEN
     MENUITEM "&Save...\t^S",  CW_FILE_SAVE
     MENUITEM "Save &As...", CW_FILE_SAVEAS
     MENUITEM "&Clip Chart...", CW_CLIP_CHART
     MENUITEM "&Print Charts\t^P", CW_FILE_PRINT
     MENUITEM "&Quit\t^X", CW_FILE_QUIT
     MENUITEM SEPARATOR
     MENUITEM "A&bout Borland...", CW_ABOUT_BORLAND
    END
```

Dialog box resources can be included in an .RC file using .RC include statements, or can be manually inserted into an .RC file. As shown here, .DLG files are in a format compatible with .RC files.

```
102 DIALOG DISCARDABLE LOADONCALL PURE MOVEABLE 77, 94, 165, 71 STYLE WS_POPUP | WS_CAPTION
BEGIN
  CONTROL "" 502, "EDIT", WS_CHILD | WS_VISIBLE | WS_TABSTOP | 0x80L, 10, 32, 138, 12
  CONTROL "" 500, "STATIC", WS_CHILD | WS_VISIBLE | WS_GROUP, 11, 5, 143, 18
  CONTROL "&OK" 1, "BUTTON", WS_CHILD | WS_VISIBLE | WS_TABSTOP | 0x1L, 32, 50, 32, 14
  CONTROL "&Cancel" 2, "BUTTON", WS_CHILD | WS_VISIBLE | WS_TABSTOP, 99, 50, 32, 14
END
```

.RC and .DLG files are useful for printed archives of your resources or for compatibility with the .RC compiler. In most cases, they're not necessary, since the Resource Toolkit lets you directly edit .EXE and .RES files, eliminating the need for script files.

Bitmap, cursor, and icon files

Bitmap (.BMP), cursor (.CUR), and icon (.ICO) files are stored in bitmap format. Each type of file can contain only a single resource of the appropriate type. For example, a .CUR file can contain only a single cursor resource. As with .DLG files, .BMP, .CUR, and .ICO files can be generated by the Resource Toolkit, and they can be included in an .RC file.

Header files

In Windows programs, resources are identified by number. For example, a set of dialog box resources in an application might have the unique identifiers 100, 101, and 102. Windows programs also use numbers to identify the action resulting from a menu selection, and to identify individual controls inside a dialog box. A typical application might use hundreds of these identifiers.

Numbers, however, are limited in meaning. They make poor names for things, because all they convey is sequence or magnitude. For this reason, Windows lets you use a header file to assign symbolic names to the numbers used in an application. The header file associates symbols with the numbers. The language compiler and the resource editor translate the symbol back into the identifying number. This lets you work with meaningful names rather than a confusing list of numbers. Here's part of a typical header file that's compatible with the Resource Toolkit and C (and C++).

```
/* D:\TEST\CHART.H */
#define  CW_MENUS              999
#define  CW_FILE_NEW           501
#define  CW_FILE_OPEN          502
#define  CW_FILE_SAVE          503
#define  CW_FILE_SAVEAS        504
#define  CW_FILE_PRINT         505
#define  CW_FILE_QUIT          508
#define  CW_CLIP_CHART         509
```

The Accelerator, Dialog box, Menu, and String editors provide access to header files and also let you create or edit the symbols defined in those files.

C H A P T E R

3

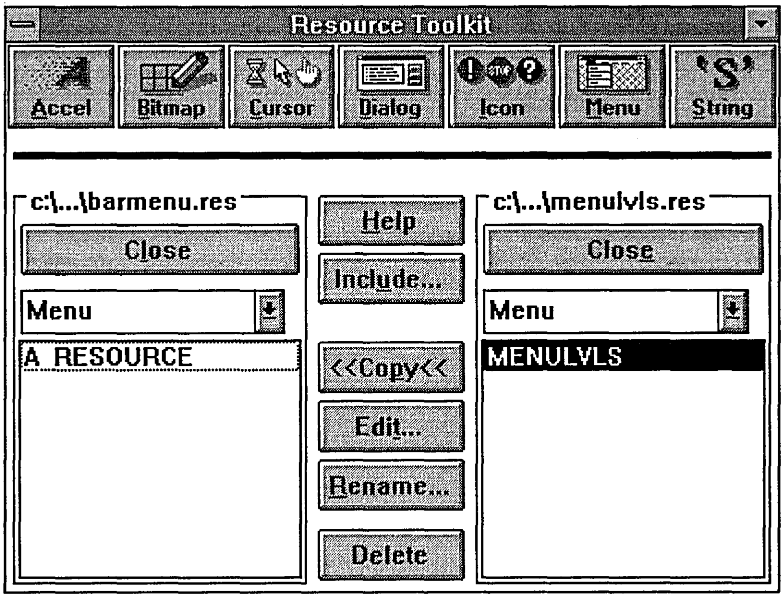
The Resource Manager

The Resource Manager is the main window and the center of all activity in the Resource Toolkit. It provides access to existing resources and to each of the resource editors. Using the Resource Manager, you can

- Start any of the Resource Toolkit's editors.
- Access existing resources so you can edit, copy, or delete them.
- Create a new resource file so you can copy resources into it.
- Close a file after you've copied or deleted resources.

The Resource Manager is designed without menus for quick operation. To activate any button, you can click the button or press the keyboard key corresponding to the button's underscored letter. For example, you can press *N* to choose the Include button. You can also press the *Tab* key to cycle to a button or field, then press *Enter* to "press" the button. The Resource Manager has three main components: the editor buttons, the resource browsers, and the include button.

Figure 3.1
The Resource Manager



Editor buttons. Across the top of the screen are buttons for each of the editors. To start an editor, click its button.

Resource browsers. The Resource Manager provides two resource browsers, each of which lets you open a file. (You can also open files within the individual resource editors.) Because there are two browsers, you can open two files simultaneously and quickly copy resources from one file into the other. This is especially useful for copying resources into an .EXE file. Alternatively, you can open one file, then delete or edit resources from that file. If you choose to edit a resource, the Resource Manager automatically starts the appropriate editor and loads the resource into that editor.

Include button. To select the resource types you want displayed when a file is open, the Resource Manager has an Include button. The Include button opens a dialog box that lists all the resource types you can edit or copy; you can select any combination of resource types to display from an open file.

Starting an editor

For information on increasing available memory, see Appendix A.

To start one of the Resource Toolkit's editors, click its button. Alternatively, press the keyboard key that corresponds to the first letter in the editor. For example, press *D* to start the Dialog Box editor. You can open as many editors as memory allows. For example, you might open an Icon editor and a Menu editor. You can also open separate copies of a single editor (for working on different resources); you could open three versions of the Icon editor, for example.

Instructions for using each editor are given in separate chapters in this manual.

Once an editor is open, you can create a new resource or edit an existing one.

Creating a new resource

You can create a new resource in one of three ways. The simplest way is to start the appropriate editor by clicking its button. If you are already within the editor, just choose File | New.

If you aren't in an editor, you can click the New button that's located above one of the Resource Manager's resource browsers. Use this method to create a new file so you can copy a resource from an existing file.

Accessing existing resources

You can access existing resources in two ways: from within an editor, and from the Resource Manager. The steps for both ways are essentially the same: You choose the resource type(s) you want to see displayed, then you open a file and select an individual resource to edit or one or more to copy or delete. These steps are accomplished somewhat differently for each method.

From within any editor, choose File | Open, choose a resource type, then select a file, open it, select a resource, and click OK.

From the Resource Manager,

1. Before opening the file, use the Include button to choose the resource types you want to see displayed from the open file.

2. After choosing the resource types to display, use one of the Resource Manager's browsers to open the file, then select one of the resource types. This displays individual resources of the selected type.
3. Once the individual resources are listed, you can select one to edit, or select one or more to copy or delete.
4. The Resource Toolkit automatically makes a backup copy of the original file, in case you want to retrieve it later.

The following sections discuss each of these steps for both the Resource Manager and File | Open in detail.

Choosing resource types to display

At any time, you can limit the resource types that will be displayed for an open file. For example, you can limit the display to only the file's icon and cursor resources, or only its menu resources.

To limit the display of resource types,

1. From the Resource Manager, click the Include button. This opens a dialog box that lists all the resource types you can edit in the Resource Toolkit. It also lists Font, RC data, and User resource types, which you can copy but not edit.
2. Check each resource type you want to include in the display, and uncheck each resource type you want to exclude.
   - Click each empty check box you want to check.
   - Click a checked box to uncheck it.

The Include Resource Type dialog box provides a quick way to include or exclude all resource types. If none of the check boxes are checked, or some of them are checked, click the Select All button to select all the resource types. If all the resources are checked, the button changes to an Unselect All button, which clears all selections when clicked. From the editor's File | Open dialog box, just set the File Type radio button.

Opening an existing file

To open an existing file (both from the Resource Manager and from within an editor), first select the resource types you want to see displayed when a file is open, as described in the previous section. Then,

1. In the Resource Manager, determine which resource browser you want to use and click its Open button. This opens the File Open dialog box. If you are opening a file within an editor, you are already looking at the File Open dialog box.

   Available files are listed in the File Open dialog box's left list box, with the current directory identified above the list box. The right list box shows the available directories.

   Note Above the list box, the dialog box indicates the types of files that will be displayed in the left list box. These file types are *not* affected by the resource types you select in the first step. The Include button affects the display of resource types from an open file. It doesn't affect the display of file types from a directory. Also, once the file is opened from within an editor, you won't be able to choose among all the resource types stored in the file.
2. If the file you want isn't in the current directory, change to the directory where it's stored.
3. Double-click the file you want, or select it and click the dialog box's OK button. In the Resource Manager, this displays a resource type in the browser's edit field, and selects the first resource of that type in the browser's list box. In the editor, this displays a Resource Selection dialog box.

   ➡ If the file isn't a valid file holding Windows resources, the Resource Manager beeps to indicate it can't open the file. For example, if you select a non-Windows .EXE file, the Resource Manager can't open it.
4. Select the resource type you want in any of the following ways:

   - For both the Resource Manager and any editor, you can press *Enter* to accept the selected resource. If you are working from the Resource Manager, this starts the resource's corresponding editor. Depending on the editor, you may get a small confirmation dialog box; for example, for image size in the icon editor.
   - In the Resource Manager, activate the browser's edit field by clicking it, or by pressing the *Tab* key until the field is activated. Then type the first letter of the resource type you

want to select. For example, you could tab to the edit field and type *D* to select dialog box resource types.

- Click the arrow to the right of the browser's edit field. This drops down a list box that displays all available resource types. To select a resource type from the list, click it, or press the key corresponding to the type's first letter. For example, you could press *D* to select dialog box resource types.

  Once you select the resource type, individual resources of that type are listed in the browser's list box.

Note For copying, renaming, or deleting, you can select multiple resources. To select a range of resources, select the first resource in the range, then press the *Shift* key and select the last resource in the range. This selects all inclusive resources as well. To select individual resources, press the *Ctrl* key and click each resource you want to select. To de-select an individual resource from the group, press the *Ctrl* key and click that resource.

5. Edit, copy, or delete the selected resource(s), as discussed in the following sections.

Editing a resource

To edit an existing resource starting from the Resource Manager,

1. Open the file that contains the resource.
2. Select a resource type from the resource browser's edit field. For example, you might select dialog box. This lists individual resources of the type in the browser's list box.
3. In the list box, double-click a resource's name or number, or select the resource and click the Edit button. This automatically starts the appropriate editor and loads the resource into that editor.

From within an editor, choose File | Open, as described previously.

Deleting a resource

To delete a resource from a file (Resource Manager only),

1. Open the file that contains the resource.

2. Select a resource type from the resource browser's edit field. For example, you might select Dialog box. This lists individual resources of the type in the browser's list box.
3. In the list box, select the resource(s) you want to delete.
4. Click the Delete button. This deletes all selected resources from the open file.

Caution! The resources are immediately deleted from the file. Make sure you select only those resources you don't want or need.

Copying a resource

To copy a resource from one file to another (Resource Manager only),

1. Use one of the resource browsers to open the file whose resource you want to copy.
2. Use the other browser to open the file you want to copy to. This can be a new or an existing file.
3. From the edit field for the source file, type or select the resource type you want to copy. Once a type is selected, individual resources of that type are listed in the browser's list box.
4. Select one or more resources to copy.
5. Click the Copy button. This opens the Resource Attributes dialog so you can rename the resource. You might want to do this if you want to replace one program's resource with another's. Once you know the name the program expects, you can use that name for the replacement resource. You can also use the attributes box to change the resources attributes. (The attributes are defined in Chapter 10.) If you selected multiple resources in the last step, the dialog opens separately for each selected resource.

Note The Copy button's arrows show the direction you can copy. The direction changes depending on which browser you use to select resources. The arrows always point *from* the selected resources *to* the other file.

6. If you don't want to rename the current resource, click the OK button. To rename the current resource, type the new name into the dialog's Name field, then click the OK button.

Renaming resources

You can change the name of any resource in an open file without having to copy it from one file to another. To do so, follow the instructions for copying the resource, except that you don't have to open a second file, and you should click the Rename button rather than the Copy button.

You can also rename a resource from within an editor by choosing File | Save As.

Backup files

Every time you save a file, the Resource Toolkit makes a backup file using the original file name but replacing the first character of the file extension with a tilde (~). To restore the file, replace the ~ with the first letter of the file extension.

For example, if you make changes to a file named CHART.ICO, then save the changes, the Resource Toolkit retains a backup of the original file, naming it CHART.~CO. To restore that file, use the DOS rename or copy command to rename the file.

Creating a new file

Use either of two methods to create a new file in the Resource Toolkit:

- Start one of the Resource Toolkit's editors and choose File | New to create the new file. Use this method to create a new file and begin defining a new resource. Instructions for starting an editor are given on page 23.
- Create the new file in the Resource Manager by pressing the New button above either of the browsers. Use this method to create a new file so you can copy resources from an existing file. As you'll see in the following paragraphs, you can copy multiple resources at once, without having to start an editor to load each resource.

To create a new file so you can copy existing resources into it:

- Determine which of the Resource Manager's resource browsers you want to use and click its New button. This opens a dialog box. Files stored in the current directory are listed in the dialog's left list box, with the current directory identified above the list box. The dialog's right list box shows the available directories.

- To create the file in a different directory, change the directory.
- Either create a new file or replace one of the existing files that are displayed in the left list box:

  To create a new file:

  - Click the File Extension radio button for the type of file you want to create. The radio buttons are to the right of the selection boxes. The types of file you can create are listed in the top right corner of the dialog box.
  - Position the cursor in the entry field and type a name for the file. Don't include a file extension; the extension is taken from the File Extension radio button.

  To replace one of the files listed in the left list box, select the file. This automatically copies the file's name into the entry field, and also selects the corresponding File Extension radio button.

Click the dialog box's New button.

If you created a new file name, this creates the file on disk and also opens it.

If you selected an existing file to replace, this opens a dialog box to confirm the replacement:

- Yes replaces the existing file with a new, empty file.
- No returns you to the dialog box so you can specify another file name.

Caution! If you click Yes, the new file is immediately created on disk, replacing the existing file. Don't select Yes unless you're sure you want to replace the existing file.

Once the file is created, its name is displayed above the resource browser. Since the file is new, the browser's edit field is empty.

To copy resources into the file, use the Resource Manager's other resource browser to open an existing file. Once both files are open, you can copy resources from the existing file into the new file.

Closing a file

When you've used the Resource Manager to open a new or existing file, the resource browser's Open and New buttons change to a Close button. To close the file when you're done with it, click the Close button.

When you click the Close button, all changes you made to the file are immediately saved. Close *does not open* a dialog box to confirm you want to save the changes.

To close a file from within an editor, choose Close from the Control-box menu in the upper left corner of the editor.

C H A P T E R

4

The Accelerator editor

The Accelerator editor lets you create and edit accelerator resources. Accelerators are hot keys for issuing an application command. Within the Accelerator editor, you can create accelerators by pressing the key rather than entering its code.

Creating accelerators

If you need instructions for creating or opening a file, see page 23.

The Accelerator editor stores accelerator keys and codes in an *accelerator table*. An accelerator table normally contains seven columns (eight if you're editing a header file) that display information about the accelerator. These columns are defined at the end of this section. If a header file is open, an additional column shows the symbolic name of the accelerator, as discussed starting on page 34.

Navigating the editor

Navigate the accelerator table with the keys discussed in Chapter 10. To add a new row to the table, press *Ctrl-Enter*. This works regardless of which field is active when you press *Ctrl-Enter*. Default values are automatically entered for each column that has a default.

Editing text

You can edit the Value field (and the Symbol field when a header file is open). The remaining fields are selection fields. You can edit these fields as discussed in Chapter 10.

The accelerator table

The accelerator table is where you actually define accelerator keys. Generally, you can define any virtual key or any character key as an accelerator in your application. However, you should avoid conflicts with Windows's use of accelerators. For example, you might not want to define *F10* as an accelerator in your application because it conflicts with Windows's use of *F10* to activate the menu bar. In addition, for better user familiarity you might want to follow the standards accepted by most Windows applications. For instance, if your application has an Edit menu, you might define the *Del* key as the accelerator for Cut, *Shift-Del* as the accelerator for Paste, and so on.

For detailed guidelines to the conventions observed by most Windows programs, see IBM's *Systems Application Architecture: Common User Access Advanced Interface Design Guide*, 1989.

The accelerator table's eight columns are: Type, Key, Code, Shift, Ctrl, Value, and Invert, and Symbol if you have a header file open. Using these columns, you can set information about each accelerator. The following text explains each column in detail.

Type

The Type column indicates whether the accelerator is defined in Virtkey or ASCII code.

Virtkey stands for *virtual key* and names accelerators according to a set of virtual key definitions defined for Windows. ASCII shows the accelerator in terms of its ASCII value. For example, the key combination *Ctrl-S* is Virtkey *S* and ASCII *^S*.

You can change the type by pressing *Spacebar* or typing a `V` or an `A` while in the Type field.

Once you've defined an accelerator, you can view it in its other form by cycling to the opposite choice; however, not all accelerators can be converted from one type to the other. For example, there's no ASCII equivalent for *F5* or the *PgUp* key, and there is no Virtkey equivalent for an exclamation point (!).

Key This column represents the *accelerator key*. To define the accelerator, press the desired key. For example, to define *Shift-Ins* as the accelerator—a common accelerator for Paste—hold down the *Shift* key and press *Ins*.

When you define an accelerator key, all fields in the row, except for Value, are automatically filled in.

Depending upon the accelerator you define, different strings may appear in this field. For example, if you define a Virtkey *Ctrl-A*, an *A* appears in this field. However, if you define an ASCII *Ctrl-A*, *^A* appears in this field.

If you try to enter an accelerator that's invalid for the format selected in the Type field, the editor beeps to indicate it can't use that key.

Code This column gives the accelerator's *keyboard scan code*. This is automatically filled in when the Key field is filled. The code, which depends on whether Virtkey or ASCII is selected in the Type column, is a key's ID number on the keyboard. It is *not* the ID used in source code to identify the accelerator; that is displayed in the Value column.

Shift This column indicates whether the *Shift* key must be pressed to activate the accelerator displayed in the Key field.

If Virtkey is selected in the Type field, this column cycles between *Yes* and *No*. The value is automatically updated when an accelerator is defined in the Key field, so it's usually unnecessary to change this field yourself. To change it, use the *Spacebar*, or press *Y* for *Yes* or *N* for *No*.

If ASCII is selected in the Type field, this field is blank. The shift key doesn't apply to ASCII accelerator codes.

Ctrl This column indicates whether the *Ctrl* key must be pressed to activate the accelerator displayed in the Key field.

If Virtkey is selected in the Type field, this column cycles between *Yes* and *No*. The value is automatically updated when an accelerator is defined in the Key field, so it's usually unnecessary to change this field yourself. To change it, use the *Spacebar*, or press *Y* for *Yes* or *N* for *No*.

If ASCII is selected in the Type field, this field is blank. The *Ctrl* key doesn't apply to ASCII accelerator codes.

Value — This column provides the ID number for the accelerator defined in the Key field. To load the accelerator into your application, refer to this number in the application's source code. Alternatively, define a symbol for the accelerator.

Enter the value as it's defined in the source code. The value must be an integer.

The ID values of accelerators can be the same as the IDs used in other resources, such as for the controls in a dialog box. This is because Windows interprets the ID value within the context of the resource currently being used in the application.

Invert — This column indicates whether the main menu item associated with the accelerator should be highlighted when the accelerator is pressed. The value must be either *Yes* or *No*.

Symbol — This column gives the symbol whose ID number in the header file matches the value for the current accelerator. To load the accelerator into your application, you can refer to this symbol in the application's source code. See Chapter 9 for a more detailed explanation of header files; the next section tells briefly how to open one.

Header files

For more information on header files and instructions on editing symbol values, see Chapter 9.

To open a header file so you can browse or edit symbols while defining accelerator resources, choose File | Open Header from the Accelerator editor's menu. This adds a Symbol column to the accelerator table. Opening a header file opens a special header-file editor. You can edit symbol values with the header-file editor or edit them directly from the accelerator table.

C H A P T E R

5

The Bitmap, Cursor, and Icon editors

The Resource Toolkit includes separate editors for each of three graphic resource types: bitmaps, cursors, and icons. This chapter describes these three editors together, since they share most operations in common.

To create a graphic resource, you use tools like those from a paint program to draw it in the editor. In addition to choosing the type of line or shape to draw, you choose the color for the graphic. For icons and cursors, you can view how that color will appear against different background screen colors.

Each of the graphic resource types is a device-independent bitmap. Each has a different size and function in an application.

Bitmap. A bitmap resource describes a picture but has no immediate window manager function. For example, a copyright dialog box might display a product's logo. The logo identifies the product, but it doesn't serve a functional purpose in the dialog box.

Cursor. Cursors are specialized bitmaps that define the location of the mouse pointer. It's useful to use different bitmaps to represent changes in the mouse's function. For example, when the mouse is used to locate an insertion position for text, the cursor might be shaped like an I-beam. When the mouse is used to move an item on the screen, the cursor might be shaped like a hand.

Icon. Icons are specialized bitmaps that represent applications or actions within an application. For example, when you minimize

Windows applications (such as the Resource Toolkit), icons are displayed onscreen to represent the minimized programs. If an application has independent windows that can be minimized, such as the editors in the Resource Toolkit, each window can have a different icon.

Files

The graphic editors can edit bitmap, cursor, and icon resources from resource (.RES) files, executable (.EXE) files, and dynamic link libraries (DLLs). They can save new resources directly into a .RES file or into an existing .EXE file. They also let you create, edit, and save bitmap (.BMP), cursor (.CUR), and icon (.ICO) files. These files can be included in the resource script (.RC) file used to compile a .RES file. See Chapter 2 for a discussion of .RES, .EXE, and .RC files.

Bitmap files

The Bitmap editor can create and edit bitmap files containing device-independent bitmaps. These can be 2-color or 16-color bitmaps whose size is only limited by available memory. Each bitmap file contains a single bitmap image, though .RES files may contain any number.

OS/2 format bitmaps

Although you can't create OS/2 format bitmaps in the Resource Toolkit, you may still be able to edit them. This ability has not been fully tested, nor is it officially supported.

Cursor and icon files

Cursor and icon files contain multiple images, each of which is designed for display on a different device. What differs among the images is their pixel dimensions and color capacity, which vary according to the requirements of different display devices. When your application loads a cursor or icon, it calls it by name. Windows then selects the appropriate image for that cursor or icon, based upon the pixel dimensions and color capacity required by the device driver.

When you create a cursor or icon, you must create each of the images that will be stored for it. To create a new image, first define its resolution by specifying the appropriate dimensions and color capacity for a particular display device. These are defined in a file called WRT.DAT, which is discussed in the following section.

Setting resolution

Resolution information for various device drivers is stored in the file WRT.DAT. This information is required for creating a new image for a cursor or icon. To guarantee that each cursor or icon you create will look good on all display devices, you can create one image for each record in WRT.DAT. However, if an image isn't defined for a particular device, Windows selects the image that will look best on the current device.

Generally, most devices use 32-by-32 pixel images, so you need one image in that dimension. You should then create a separate image for those devices that don't use 32-by-32 pixel images. For example, CGA devices use 32-by-16 pixel icons, so you should create an icon in that dimension if your application will support CGA devices. If you create only 16-color images, monochrome devices will dither all colors except black and white. (See page 40 for information on dithered colors.) When you first install the Resource Toolkit, WRT.DAT looks like this:

```
4-Plane,16,32,32,32,32
3-Plane,8,32,32,32,32
Monochrome,2,32,32,32,32
CGA,2,32,32,32,16
```

Each record in the file specifies display information for a different device. The fields in the record, separated by commas, represent the following information:

- *Name.* The arbitrary name used in the file to refer to the display device. The name can be up to ten characters long.
- *Number of colors.* Number of colors for the cursor or icon image.
- *Cursor width.* Width of a cursor, in pixels.
- *Cursor height.* Height of a cursor, in pixels.
- *Icon width.* Width of an icon, in pixels.
- *Icon height.* Height of an icon, in pixels.

Each field in the record is terminated by a carriage return. A null character cannot be used to terminate the record. The file can contain comments. Each comment line must begin with a semicolon (;) in the first column.

Cursor and icon editors

If you need instructions for creating or opening a file, see page 23.

To create a new *image* for an *existing* icon or cursor, choose New from the Image menu. To create a *new* icon or cursor resource, choose New from the File menu. Either choice opens a dialog box that lets you specify a resolution for the image. Available resolutions can be displayed in the dialog box and are read from file WRT.DAT, which is discussed on page 37. You'll want to create one image for each record in file WRT.DAT. Initially the dialog box displays the resolution for the image you've been editing; by default, cursors are 32-by-32 pixels and two colors and icons are 32-by-32 pixels and 16 colors.

If you've been editing an image and are now creating a new image, it displays information for the image you've been editing. To load an alternative image, choose Open from the Image menu. This opens a dialog box so you can select the image.

The bottom half of the dialog box has an edit field that lets you define the image resolution for the resource. Initially, the edit field displays the first record from WRT.DAT. To accept that resolution, click OK or press *Enter.*

To select an alternative resolution,

1. Click the arrow to the right of the edit field that displays the current resolutions. This drops down a list box that displays all the available resolutions from WRT.DAT.
2. Select the resolution you want and click OK.

Note You can't change the resolution changed after you begin working on the image.

Bitmap editor

To create a new bitmap, choose New from the File menu. This opens a dialog box that lets you define the height and width of the new bitmap and set the number of colors. By default, the dialog specifies a 16-color bitmap with dimensions of 72-by-72 pixels.

You can select a two-color bitmap, and/or change the dimensions for the height and width. The values you can specify for both height and width start at 1 pixel and go up to any size, limited only by available memory.

Note You can't change the bitmap's size and color after you begin working on the image.

Using a graphics editor

The graphics editors provide features common to most paint programs:

Color palette. On the left is a color palette for choosing the colors you want to work with. Above the palette is a color box showing the currently selected color. For the Icon and Cursor editors, there are three color modes:

- *Color* selects colors that won't change, regardless of the background screen color.
- *Screen* selects colors that will match the screen background and change if the screen background color changes.
- *Inverse* selects colors that always reflect the inverse color of the background screen color.

Before selecting a color from the palette in these editors, select the color mode that the palette choice affects. The color palette is discussed in detail starting on page 41.

Tools palette. Across the top of the editors is the tools palette. You'll use these tools to draw the bitmap, cursor, or icon. The tools palette includes a toggle switch for toggling colors automatically, a line width tool for setting the thickness of line, and a set of drawing tools. The tools palette is discussed in detail in the section "Drawing and editing graphics," starting on page 44.

Editing area. The editing area is where you draw the bitmap, cursor, or icon. The editing area can provide a magnified view of the graphic, making it easier for you to work on it.

Rulers. Abutting the left and top sides of the editing area are two rulers that show the exact location of the drawing tool.

View window. To the right of the editing area is a small view window that shows the entire image as it will appear in the application. The View window always mirrors what's in the editing area.

In the Bitmap editor, the View window is the exact size of the bitmap. In the Cursor and Icon editors, the View window is larger than the actual image to show how the image will appear against various background colors. To see what part of a large resource is currently in the editing area, move the mouse pointer into the View window, then click and hold down the left mouse button.

Resource statistics. In the bottom right corner of the editor is an area that shows the number of colors allowed for the graphic, and the graphic's height and width in pixels.

Using color

Before you begin drawing with a selected tool, choose the color you want for the new line or shape. For a cursor and icon, determine whether you want to draw in Color mode, Screen mode, or Inverse mode. The distinctions between these modes are discussed in the next section. The color palette shows the available colors for the current resource. The palette uses two types of colors: pure colors and dithered colors.

Pure colors are colors that can be directly generated by the output device. On some devices, such as VGA monitors, the first 16 colors—both columns of colors from the first eight rows of the palette—are pure colors. Pure colors are combined values of red, green, and blue (RGB) that are guaranteed to be distinct on devices that support 16 or more colors. For that reason, all lines drawn by the graphic tools use pure colors. If you select a dithered color for drawing a line, the editor uses the closest pure color instead.

Dithered colors are colors that the windowing environment synthesizes because the output device can't generate them directly. This is done by a process known as dithering, where pure colors are combined in a pattern of dots to approximate another color. The simplest example is creating gray by dithering black and white. On VGA devices, the first eight rows in the color palette are pure, and the remaining colors are dithered.

Dithered colors can bring a variety of color to your images. Because they're simulated by a series of dots, they don't have good resolution for drawing lines and are best used for coloring areas.

Dithered colors have several disadvantages. For example, though they may look good on your display device, they won't necessarily look good on all devices. Moreover, for some display devices, the windowing environment might find it necessary to switch a dithered color to a pure color, and the pure color might not blend well with other colors in the image.

➡ If you're not sure whether a color is pure, use the Pouring tool to fill a large area with it. If the color in the area appears solid, it's pure. If it has a checkered pattern, it's dithered.

Color palettes

The color palette for the Bitmap editor has 28 colors. If you specify a 16-color bitmap, the first 16 are true colors, the rest are dithered colors. If you specify a 2-color bitmap, the palette contains black, white, and 26 shades of gray.

Because Windows requires that all cursor resources be black and white, the palette for the Cursor editor contains only black and white for drawing. However, 16 screen and inverse colors are still available in the lower palette. On some devices, some of these colors may be dithered.

The number of colors in the Icon editor's palette is determined by the number allowed by the device selected from WRT.DAT when the image was created. For some devices, the Icon editor has a single color palette, and you can select any color from the palette for all three color modes (defined in the next section). For other devices, the icon editor has two palettes. The top palette provides the colors available for drawing. The bottom palette provides 16 screen and inverse colors; on some devices, some of these colors may be dithered.

Choosing color for cursor and icon images

The Icon editor and Cursor editor provide three selections for color drawing mode: Color, Screen, and Inverse. Only one mode at a time can be in effect; it's indicated by a heavy border around its button. You select the mode you want by clicking the appropriate button. The color for each mode is displayed in a corresponding color box.

Color

Color specifies a permanent color that retains its hue, regardless of the background screen color.

Screen

Screen specifies the color of the background screen. This color mode lets you see how your image will look against various background colors. The current color for Screen mode is displayed as a background color in the View window.

Any lines you draw in the image with Screen mode will always take on the color of the screen background, making them appear invisible in the application. You can therefore use screen mode to suggest a transparent area.

Inverse

Inverse specifies the color that is the inverse of the screen color. When a line or shape has been drawn in Inverse mode, it always takes the color that's the inverse of its background on a display device.

Permanent colors are invisible when the permanent color is the same color as the background screen color. Inverse colors, on the other hand, change color as the screen color changes. Thus, Inverse colors are always visible against any screen background color. To ensure an image is always visible against any screen background, you can outline it with a color in Inverse mode.

The Screen and Inverse color modes always change together. For example, if you change the Screen color, this automatically changes the color for Inverse mode, and also changes the color of any lines in the editing area that were drawn with Inverse mode.

Customizing color palettes

You can customize the Bitmap editor color palette by changing any of its colors. The hue for a color is always a mixture of red, green, and blue (RGB).

16-color bitmaps

To edit any color from the palette in a 16-color bitmap,

1. Double-click the color in the palette. Alternatively, select the color, then choose Edit Color from the Palette menu. This opens a dialog box that lets you change the color's RGB color number.

2. If you know the RGB color number you want, you can type it into the edit fields. Each field accepts a range of values from 0 to 255. If you don't know the RGB color number, you can use the scroll bars to scroll the numbers up and down.

   If you use the edit field, the corresponding scroll position is automatically updated. If you scroll, the number in the corresponding edit field is automatically updated. The color box to the right of the scrolls always shows the current blend of red, green, and blue.
3. Click Accept, Cancel, or Default:
   - **Accept** uses the color you specified to replace the original color in the palette.
   - **Cancel** cancels the editing operation and restores the original color to the palette.
   - **Default** selects the palette position's default color value. Every position in the palette has a default color. That color may or may not be the original color you selected.

2-color bitmaps

To edit the text or background color for a 2-color bitmap,

1. Choose either Text Color or Background Color from the Options menu. This opens a dialog box that lets you change the color's RGB color number.
2. If you know the RGB color number you want, type it into the editing fields. Otherwise, use the scrollbars to scroll a color blend. The color box to the right of the scrolls shows the current blend of red, green, and blue.
3. Click Accept or Cancel.
   - **Accept** uses the color you specified to replace the original color in the palette.
   - **Cancel** cancels the editing operation and restores the original color to the palette.

Note With a 2-color bitmap, you can't save the modified palette as you can with a 16-color bitmap. However, saving the bitmap into a file also saves its color values. When you later open the bitmap file, the palette contains the edited rather than the default text and background colors.

Saving a customized color palette

To save an edited color palette,

- Choose Save Colors from the Palette menu. This opens a dialog box so you can name the new palette.
- Specify a directory and name for the new palette. The file extension must be PAL.

Retrieving a customized color palette

At any time, you can get a customized color palette for use during an editing session:

- Choose Get Colors from the Palette menu. This opens a dialog box so you can specify an existing palette.
- Specify a directory and name for the palette, or select them from the list boxes. Only files with file extension PAL are available.

Drawing and editing graphics

To draw and edit graphics, use the tools from the Tools palette. All three editors provide identical tools. The Cursor editor has a special tool for defining the cursor's hot spot.

Using the graphic tools

The graphic tools, shown in the following figure, provide a wide range of capabilities.

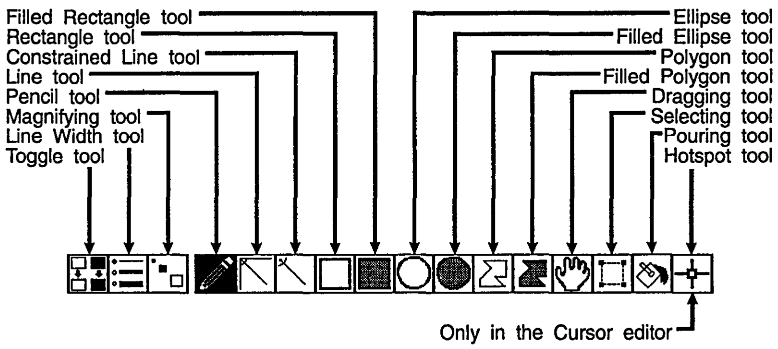


Figure 5.1
The graphic tools

The procedure is the same for using any graphic tool:

1. Select the color you want to work with. For cursors and icons, you can select a Screen or Inverse color in addition to a permanent Color. Color considerations are discussed in the section "Using color" on page 40.
2. Choose a line thickness with the Line Width tool, and choose the editing area's magnification with the Magnifying tool.
3. If you want the Pencil tool to toggle to inverse colors, turn on the Toggle tool.
4. Select the tool you want to use. By default, the Pencil tool is selected. To select a tool, click it in the Tools palette.
5. Position the pointer where you want to begin drawing.
6. Click the left mouse button. Holding the button down, drag the mouse in any direction. This extends a line or rectangle, depending on the tool you've selected. As long as you keep the left mouse button depressed, you can continue to adjust the size of the line or shape.
7. Release the mouse button when you have the size you want. This draws the line or shape, using the position and size you specified. For a polygon, repeat the drawing steps for each side of the line.

Toggling inverse colors

The Toggle tool changes only color values. It doesn't change the drawing mode to Inverse mode.

It's easy to choose a color from the palette and begin drawing with it. But what if you want to draw with a particular color's inverse value? You can use the Toggle tool to automatically toggle inverse colors. The Toggle tool works only with the Pencil.

By default, the Toggle tool is off. To turn it on, click it. To turn it off again, click it again. The tool indicates its current status through two sets of rectangles. When the toggle is off, the rectangles suggest movement from white to white and from black to black, indicating the color won't toggle. When the toggle is on, the rectangles suggest movement from white to black and from black to white, indicating the color will toggle.

The color toggles with each mouse click. When you click the mouse button, the Pencil reads the color value of the pixel at the current cursor position and paints with the inverse color. Thus, when you begin drawing over a white area, the Pencil draws in black. When you begin drawing over a black area, the Pencil draws in white. It always draws with the inverse color to the starting pixel's color value.

The mouse button must be released for the tool to toggle between colors. Thus, if the Pencil is currently drawing in white and you drag it over a white area, it doesn't toggle to black.

Choosing a line width

Before using any drawing tool, use the Line Width tool to select a line width. The Line Width tool measures all widths in pixels and provides three thickness:

- Thin line: 1 pixel
- Medium line: 3 pixels
- Thick line: 5 pixels

By default, the Line Width tool draws a thin line. To select another thickness, click the tool, or use the Options menu to choose a width. Each time you click the Line Width tool, the width cycles to the next selection. The tool indicates the current selection by filling the bullet that's to the left of the selected width.

All lines in the editors are drawn with pure color, even when you choose the thick line width.

Magnifying the editing area

Before drawing, use the Magnifying tool to select a scale for the editing area. The Magnifying tool provides three options:

- Actual size
- Magnify four times actual size
- Magnify eight times actual size

By default, the Magnifying tool magnifies the editing area to four times the actual size of the graphic. To select another magnification, click the tool, or use the Options menu to choose a magnification. Each time you click the Magnifying tool, the scale of the editing area cycles to the next selection. The tool indicates the current scale by filling the square that represents the current editing area.

With four- and eight- times magnification, the entire graphic may not be visible in the editing area, though you can see it in its entirety in the View window.

Viewing a graphic

When the editing area is magnified to four to eight times the size of the actual graphic, the graphic is sometimes larger than can be displayed in the editing area. When this is the case, you can view the graphic's position in any of the following ways:

- Use the horizontal and vertical scroll bars to change the view of the editing area.
- Use the Dragging tool to drag the graphic. To use the Dragging tool, select it, then click anywhere in the editing area and drag in any direction. This has the same effect as but is faster than using the scroll bars.
- Click the left mouse button in the View window and hold the button down. The dark area represents the visible work area in the editing area. To change the view, click the right mouse button in the View window. The editing area scrolls so the point where you right-clicked is moved as close as possible to the center of the editing area.

Free-form lines and shapes

Use the Pencil tool to draw free-form lines and shapes. Like a physical pencil, a digital pencil draws a line anywhere you drag it. It can also fill in points and areas. To draw a point with the Pencil, click the desired spot without dragging the mouse. For filling large areas, use the Pouring tool.

Straight lines

Use the Line tool to draw straight lines. The Line tool draws in any direction. To draw a line that's at a multiple of 45 degrees, use the Constrained Line tool.

Constrained lines

Use the Constrained Line tool to draw a line that's vertical, horizontal, or at any multiple of 45 degrees.

Rectangles

Use the Rectangle tool to draw a hollow rectangle. Once the rectangle is drawn, you can leave it hollow, or use the Pencil or Pouring tool to fill it with a color that's different from the color you used to draw it. To fill a rectangle with the same color as you used to draw it, it's quicker to draw it with the Filled Rectangle tool.

The Filled Rectangle tool draws a solid rectangle in the selected color. You can either leave the rectangle filled in, or select another color and draw over selected sections.

Ellipses Use the Ellipse tools to draw hollow or filled ellipses. Considerations for choosing the hollow versus the filled tool are the same as those for drawing a rectangle.

Note As you drag the mouse to draw the ellipse, its shape seems perfect. When you release the mouse button in a magnified editing area, the ellipse suddenly becomes jagged. This is because the editing area shows a magnified view of the graphic, and each pixel maps to a square on your screen. When the graphic is resolved down to its actual size, the jagged lines appear more curved. If you look in the View window, you'll see that the shape looks closer to an ellipse.

Polygons Use the Polygon tools to draw hollow or filled polygons. Considerations for choosing the hollow versus the filled tool are the same as those for drawing a rectangle.

Unlike rectangles and ellipses, which can be drawn with a single click and drag of the mouse, you have to click, drag, and release the mouse button for each side of the polygon. However, to draw the final side, you can double-click the mouse button and let the tool draw it for you. In fact, if you don't close a polygon, the tool automatically completes it for you.

Dragging tool Use the Dragging tool to drag the graphic. To use the Dragging tool, select it, then click anywhere in the editing area and drag in any direction.

Selecting a graphic region Use the Selecting tool to select a section of the graphic to cut or copy to the Clipboard. Once in the Clipboard, you can perform any of the operations available from the Edit menu. The Edit menu is discussed in Chapter 10.

The Hot Spot tool In addition to the tools just discussed, the Cursor editor has one unique tool. This is the Hot Spot tool. Every Windows cursor has a unique pixel that precisely identifies the one point at which cursor operations occur. This point is called the hot spot. Regardless of a cursor's shape, it must have one pixel that can be mapped to the hot spot. Windows uses the hot spot's location to inform an application of pointing activity.

To designate a hot spot,

- Select the Hot Spot tool from the Tools palette.
- Click the desired location.

Filling large areas

Use the Pouring tool to fill large areas with color. By default, the Pouring tool floods the entire editing area with color. This makes it easy to provide a background color for a bitmap. To fill the editing area, select the Pouring tool and click any open area.

You can also fill a bounded area with color. To do so, select the color, then click the tool anywhere within the area you want to fill. When you click the mouse button, the Pouring tool reads the color value of the pixel at the current cursor position and pours the selected color into all surrounding pixels with that value. It stops pouring color when it encounters a boundary whose color value is different from the color value it initially read. This makes it easy to fill irregular areas with color.

For example, assume you draw two hollow polygons of different sizes, positioning the smaller one inside the larger one. If you click the Pouring tool in the area between the two, the area between their borders is filled. The areas outside the larger polygon and inside the smaller polygon are unaffected.

Menus

This section outlines the Image, Options, and Tools menus in the graphic editors. The File and Edit menus are described in Chapter 10; the Palette menu (Bitmap only) lets you get, save, or edit color palettes, as discussed in the section "Customizing color palettes" on page 42.

Image menu

The Image menu is available only for the Cursor editor and the Icon editor. It provides the following options:

- **New:** Creates a new image.
- **Open:** Opens an existing image.
- **Save:** Saves an image on disk.
- **Delete:** Deletes an image.

Options menu

The Options menu for all three editors lets you set the line width for the graphic tools. The widths are identical to those offered by the Line Width tool. The Options menu also lets you turn the Toggle tool on and off, and choose the magnification for the editing area.

The Bitmap editor provides one additional option: Combination mode. This option opens a dialog box that lets you determine how colors from Clipboard bitmaps will be pasted to bitmaps in the editing area for pasting.

By default, Combination mode pastes bitmap colors exactly as they're stored in the Clipboard. To change pixel color values as they're pasted from the Clipboard to the editor, change the combination of values defined in the dialog box. For example, if you reverse the settings of all the dialog box's radio buttons, the bitmap will be pasted with the inverse colors to those stored for it in the Clipboard.

Tools Menu

The Tools menu can be used as an alternative to using the tools on the Tools palette. Choosing a tool from the menu has the same effect as clicking the tool's icon. The tool descriptions starting on page 44 apply to the menu as well.

C H A P T E R

6

The Dialog Box editor

The Dialog Box editor gives you a visually-interactive method for designing dialog box resources. A dialog box is a special window containing controls, such as list boxes, push buttons, and edit fields. Controls are the elements that compose the dialog box's user interface. Using the Dialog Box editor is like using a drawing program. You create and edit dialog boxes by selecting, locating, and modifying their controls.

Files

The Dialog Box editor can edit dialog box resources from resource (.RES) files, executable (.EXE) files, and dynamic link libraries (DLLs). It can save a dialog box resource directly into a new or existing RES file, and it can also save the same dialog box specification into a dialog box resource script (DLG) file. For a discussion of these files, see Chapter 2. The Dialog Box editor can also open a header file to correlate symbols from the file with the dialog box resource.

Header files

To open a header file (also called an include file) so you can browse or edit symbols while defining dialog box resources, choose File | Open Header from the Dialog Box editor's menu.

Opening a header file opens a special header-file editor so you can edit symbol values in the header file. For more information on header files and instructions on editing symbol values, see Chapter 9.

An overview of the editor

The Dialog Box editor has two palettes:

If you need instructions for creating or opening a file, see page 23.

- The Tools palette lets you create a dialog box and all its controls. The Tools palette is discussed starting on page 59.
- The Alignment palette lets you change the alignment of the controls within the dialog box. The alignment palette is discussed starting on page 67.

By default, the palettes are located above the editing area, with the Tools palette to the left of the Alignment palette. When you change the size of the Dialog editor on your desktop—particularly when you reduce its size—you may cut off your view of the palettes. In these instances, you can change the palette positions:

- To position the palettes below the editing area, choose Palettes on Bottom from the Dialog menu.
- To position the Alignment palette to the left of the Tools palette, choose Swap Palettes from the Dialog menu.

Dialog Boxes and controls

See pages 59 and 67 for descriptions of each of the tools used in doing these tasks.

In the Dialog Box editor, you work with dialog boxes and controls to

- create them
- choose them
- move them
- change their size
- set their attributes
- set the tab order for controls
- define groups for controls

Dialog boxes with menus

Dialog boxes with menus won't display the menus in the editor. The client area of the dialog box will appear to be larger than it really is in the *y* dimension because of the space being reserved for the (non-displaying) menu.

Creating a dialog box or control

To create a new dialog box or control,

1. Select the appropriate tool from the Tools palette. For example, to create a button, choose the button tool. The tools are defined in the sections "Dialog Box tool" and "Control tools" on pages 60 and 61, respectively.

   ➡ Remember, if a tool is a multi-style tool and it doesn't display the option you want, double-click it until the icon changes to the desired option. (Double-clicking changes the tool and also selects it.)

See page 56 for information on setting attributes.

2. Position the pointer where you want to begin drawing. For a button, the position you choose determines where the button will be displayed in your application. You can see the coordinates of this position in the Dialog Attributes dialog box.
3. Click the left mouse button and hold the button down.
4. Keeping the button pressed, drag the mouse in any direction. This forms a rectangle to show you the size of the item you're creating. As long as you keep the left button pressed, you can continue to adjust the rectangle's size. However, you can't drag the mouse beyond any border of the window.
5. When you have the size rectangle you want, lift your finger off the left mouse button. This draws the item, using the size and position you specified.

Note After you create a control, the Tools palette automatically resets itself to the Pointer tool. To create another control of the same type, you can either select the tool again, or hold down the *Ctrl* key as you click and drag the mouse to create the next control. Hold down the *Ctrl* key to automatically select the last tool used.

You can use this technique to create a control without actually drawing it. Position the pointer where you want the control, press

Ctrl, and quickly click the left mouse button. This creates the control with a default size.

Selecting a dialog box or control

Before moving or sizing items in the Dialog Box editor, you must first select them. When an item is selected, you'll see a sizing box in each of its four corners. To select a dialog box, click the cursor within the dialog box. Be sure to click on a position that isn't occupied by a control.

To select a control, click the cursor anywhere within the control. Be sure the cursor is well within the control. If it's too close to one of the control's borders, you may inadvertently select the dialog box instead.

Selecting multiple controls

The Dialog Box editor lets you choose multiple controls so you can use the alignment tools to adjust their positions, or collectively move them.

To choose multiple controls, choose the first control in the regular way, then hold down the shift key and click each additional control you want to select.

Alternatively,

- Position the cursor near the corner of the set of controls you want to select.
- Click and drag the mouse until you form a rectangle around all controls you want to select.
- Release the mouse button.

De-selecting a dialog box or control

Any one of the following actions de-selects all items that are currently selected:

- Clicking the mouse button outside of the dialog box, but within the editing area.
- Choosing another item.
- Creating another control. After it's created, the new control is automatically selected.

Moving a dialog box or control

When you move a dialog box, all its controls move with it. Each control remains in its relative position within the dialog box. When you move a control, only the selected control moves. If multiple controls are selected, they all move together, maintaining their positions relative to each other.

To move a dialog box,

- Position the cursor on any of the dialog box's borders. The cursor changes from an arrow to a hand when it's on the border.
- Click and drag the dialog box to its new location.
- Release the mouse button.

To move a control,

- If you want to move multiple controls, select all the controls you want to move.
- Position the cursor inside the control (or one of the selected controls), then click and hold down the mouse button. The cursor changes from an arrow to a hand when you click the mouse button.
- Drag the control to its new location, then release the mouse button.

Changing the size of a dialog box or control

When you change a dialog box's size, the sizes of its controls don't change. Each control retains its current size and position. Thus, the minimum size you can make a dialog box is determined by the number and position of controls within it.

When you change a control's size, only the selected control is affected. If you select multiple controls, they all change size proportionately.

To change an item's size,

- Select the item you want to size. To size multiple controls, select each control. A sizing box appears in each selected item's corners.
- Position the cursor over one of the sizing boxes. When correctly positioned, the cursor changes to a cross with arrow heads.
- Click the left mouse button and hold it down.

- Keeping the button pressed, drag the mouse in any direction. This shows a rectangle of varying size. As long as you keep the left button pressed, you can continue to adjust the size of the rectangle.
- When you have the size rectangle you want, lift your finger off the left mouse button.

Restricting mouse movement

You can restrict the movement of the mouse to only one dimension when you

- create a new dialog box or control
- change the size of a dialog box or control
- move a dialog box or control

To restrict mouse movement to a single dimension, hold down the *Shift* key, then drag the mouse either horizontally or vertically. The direction you choose restricts the mouse movement to that direction until you release the *Shift* key.

Setting attributes for a dialog box or control

Attributes are the characteristics defined for a dialog box or control. Each dialog box and control in the Dialog Box editor has a corresponding attributes box. Through the attributes dialog boxes, you can

- assign a title to a captioned dialog box, or text to a control
- change the ID number of any control
- change the position or size of the dialog box or any control
- access a style dialog box for choosing additional characteristics

To define attributes for dialog boxes or controls,

To bypass the attributes box and go directly to the style dialog box, press the Alt key when you double-click.

1. Open the attributes dialog box that corresponds to the dialog box or control, using either of two methods:
   - Double-click inside the dialog box or control. For a dialog box, be sure to double-click over a position that isn't occupied by a control.
   - Select the dialog box and choose Attributes from the Dialog menu, or select the control and choose Attributes from the Controls menu.
2. Edit the attributes you want to change.

3. To advance to the styles screen for a control, click the Styles button. The styles vary according to the control whose attributes you're defining.

Assigning text

You can use the attributes dialog box to assign a title to a captioned dialog box, or text to a control. For a dialog box, use the Title entry field to assign text to a captioned dialog box. For a control, use the Text entry field to assign text to the control. The different types of controls display the text in different places:

- **Buttons.** The Text field determines the text displayed within or beside the button.
- **Edit fields.** The Text field determines the default text displayed when the dialog box first appears.
- **Static text.** The Text field determines what text is displayed in the static field.
- **Group boxes.** The Text field is used to specify the title for the group.

For remaining controls, you can assign text, but the text isn't displayed in the application.

Assigning fonts to text

You can assign a font to the text displayed within the dialog box. The font applies to all controls within the dialog box. The font doesn't have to be the same font used by other dialog boxes within your application. (However, consistency and other design considerations usually require a minimum of font changes both within a dialog box and across dialog boxes.)

To assign fonts to the text within a dialog box,

1. Open the Dialog Attributes dialog box as described on page 56.
2. Click the Font button on the attributes box. This opens a dialog box that has two combo boxes for setting fonts: one determines the font type, the other determines the point size.
3. To change the font type, click the arrow to the right of the Fonts combo box. This opens a drop-down list box so you can select a font that's installed on your machine. Alternatively, type a font type in the combo box's edit field. The font you specify doesn't have to be installed on your machine or listed in the list box.

4. Follow the same procedure for specifying a point size in the Point Size combo box.

Changing coordinates

In the Dialog Attributes box, the (x,y) field shows the *x*- and *y*-coordinates of the dialog box's upper-left corner. These coordinates represent the dialog box's position within the window that displays it in your application. They also correspond to the dialog box's position within the Dialog Box editor. The (cx,cy) field shows the dialog box's width and height.

For an explanation of dialog box base units, see Charles Petzold's Programming Windows.

In the attributes box for controls, the (x,y) field shows the coordinates of the control's upper-left corner within the dialog box. The (cx,cy) field shows the control's width and height. A horizontal unit in the (x,y) or (cx,cy) fields is $\frac{1}{4}$ of the dialog box base width unit. A vertical unit is $\frac{1}{8}$ of the dialog box base height unit. The current dialog box base units are calculated as a function of the width and height of the current system font.

The Dialog Box editor automatically calculates the (x,y) and (cx,cy) coordinates for you, based on the item's current position.

- You can change an item's position by changing the numbers in these fields. You must specify integer values.
- Alternatively, move the item in the Dialog Box editor by dragging it or using the alignment tools to adjust control positions. This automatically updates values in the (x,y) and (cx,cy) fields.

Changing control ID numbers

When you save a dialog box with File | Save As, a Resource Dialog Box opens so you can assign a name or number to the dialog box resource. This is the name or number used to identify the dialog box in source code. Controls, however, are identified by the ID number defined for them in the Item ID field in their attributes boxes.

For a new resource, the Dialog Box editor automatically assigns ID numbers to each control you create. The controls are numbered consecutively as they're created, using positive integers that begin with 101. For an existing resource, the controls already have ID numbers; new controls you create are assigned an ID number that's one higher than the highest number for existing controls.

You can change the control's ID number in the Item ID field. To show the ID numbers for all controls in the dialog box, use the Dialog menu. The Dialog menu is discussed on page 70.

The ID number for each control must be unique within the dialog box. However, the IDs can be the same as those used in the other resources—including other dialog boxes defined for the application. For example, two different dialog boxes in the same application can use ID numbers 101, 102, and 103 for their controls, and a menu resource in the application can use these numbers for its menu items. This is because Windows interprets the ID within the context of the resource currently being used in the application.

Defining styles

When the Styles dialog box opens, an item's current styles are selected in the check boxes and radio buttons. You can change the styles by selecting alternative buttons. For example, you can change a radio button to a check box by selecting Check Box in the Styles dialog box. Or you can change a captioned dialog box to a standard dialog box by selecting Standard Dialog Box from the Dialog Box Attributes box.

Setting tab order for controls

The tab order determines the order the focus moves among controls in an application when you press *Tab*. To set the tab order for controls, use the Controls menu. The Controls menu is discussed on page 71.

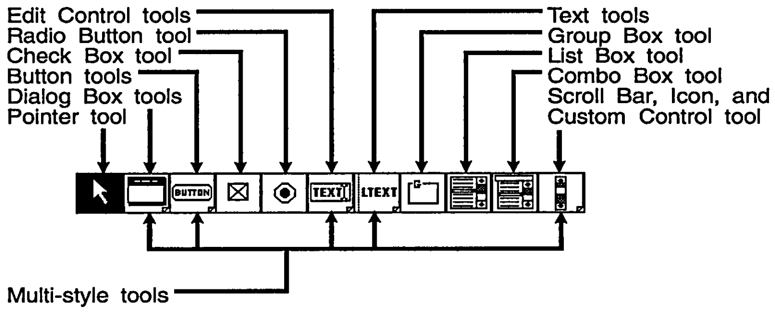
Defining logical groups for controls

A logical group defines a set of controls within which you can move input focus by pressing the arrow keys. To define logical groups, use the Controls menu. The Controls menu is discussed on page 71.

Tools palette

You can select tools using either the Tools palette or the Tools menu. Figure 6.1 shows the Tools palette. To select a tool, click it in the palette, or open the Tools menu and choose it by name.

Figure 6.1
Tools palette



Pointer tool

After you select a tool, the Pointer tool is automatically activated. The pointer lets you draw with the selected tool, or choose the dialog box or one or more existing controls so you can move them or change their size.

Dialog Box tool

The Dialog menu is discussed on page 70.

Use the Dialog Box tool to create either of two types of dialog boxes:

- a standard dialog box
- a captioned dialog box

By default, a captioned dialog box is automatically drawn when you create a new file in the Dialog Box editor. If you want a standard dialog box, you can change the dialog box's style in its attributes box, as discussed in "Dialog boxes and controls" starting on page 52. Alternatively, you can double-click the Dialog tool. This automatically changes the style of the dialog box currently displayed in the editing area.

Once you've created a dialog box, you can use the Dialog Attributes dialog box to change its style. For example, a standard dialog box can be changed to a captioned dialog box by selecting the Caption radio button in the Dialog Attributes dialog box.

Once a dialog box is created, you can create its controls. You can also use the Dialog Box menu to

- show the control tab order
- show each control's ID number
- show the logical groups among controls

■ set grid resolution

Standard dialog box Use the Dialog Box tool to create a standard dialog box. A standard dialog box is a dialog box that doesn't have a title bar. Create a standard dialog box when you don't want a dialog box to be movable within the application.

Captioned dialog box Use the Captioned Dialog Box tool to create a captioned dialog box. A captioned dialog box has a title bar with a caption in it. Create a captioned dialog box when you want the dialog box to be movable within the application.

Control tools

The tools to the right of the Dialog Box tool on the Tools palette are the control tools. Use these control tools to define all of the following elements of a dialog box:

push buttons	group boxes
check boxes	list boxes
radio buttons	combo boxes
edit fields	scroll bars
static text	icons

Multi-style tools

You've probably already noticed that the bottom right corner of several tools in the Tools palette are folded. This means that they are multi-style tools—that is, they can create more than one style of dialog box or control. For example, the Button tool is a multi-style tool because it can create different kinds of buttons. The following are multi-style tools:

dialog box tool	text tool
button tool	scroll bar tool
edit tool	

To cycle through a multi-style tool's options, double-click the tool. This changes the tool and also selects it.

Push buttons

Push buttons (also sometimes referred to as *command buttons*) trigger an immediate action, without maintaining an on/off status. The Resource Toolkit lets you create standard and default push buttons.

Standard push button

Use the Button tool to create a standard push button. A standard push button has to be selected before it can be activated. Create a standard button for each action that isn't the dialog box's default action. The Button tool is selected by default. This means you don't have to select the Button tool before you begin dragging the mouse.

Default push button

Use the Default Button tool to create a default push button. A default push button is one which is automatically selected when your application's dialog box is open. To activate it, you usually press *Enter.* You should define only one default push button for each dialog box.

Check boxes

Use the Check Box tool to provide one or more options that can be turned on or off. Each selection offered by a check box is independent and can be turned on or off. The decision of whether the edit control should allow multiple lines of text entry, for instance, is independent of whether it should contain a horizontal scroll bar.

To enclose check boxes so they appear as a visual group on the screen, use the Group Box tool.

Radio buttons

Use the Radio Button tool to provide a set of options, only one of which can be selected. All the radio buttons from a set must be defined as a logical group. To define controls as a logical group, choose Start Group from the Controls menu. The Controls menu is discussed on page 71.

To enclose the radio buttons so they appear as a visual group on the screen, use the Group Box tool.

Customizing buttons

Use the Owner Draw tool to create a button whose behavior is different from the standard behavior of push buttons, check boxes, and radio buttons, you can create an *owner-draw button*. The advantage of creating owner-draw buttons is that they're not limited to displaying text. For example, the buttons you press in the Resource Manager to start an editor are owner-draw buttons that display bitmaps.

With owner-draw buttons, the application takes responsibility for the button's appearance. When the button is selected, the parent window is "notified" and also receives a request to paint, invert, or disable the button. You need to define the additional support for these functions in the application's source code. For more information on owner-draw controls, see Microsoft's Software Development Kit.

Edit controls

Edit controls are used primarily for text entry. They can also display default text in the edit field. The Resource Toolkit lets you create the following styles of edit control:

- single-line edit
- multiple-line edit
- multiple-line edit that allows vertical scrolling
- multiple-line edit that allows vertical and horizontal scrolling

Static text

Use the Text tool to display static text. For example, the text might mention the consequences of an action or ask a question. Or it might label an edit control.

The Text tool provides three formatting options: left-justified text, right-justified text, and centered text.

Group box

Use the Group Box tool to draw a titled border around a set of push buttons, check boxes, or radio buttons so they're displayed as a visual group.

Group boxes don't need to have the group bit set; they exhibit the grouping behavior without it. Because of this, Start Group in the Controls menu is always shown as checked and disabled for group boxes, and the Group check box in the group box's Style dialog has no effect. The Controls menu is discussed on page 71.

List boxes

Use the List Box tool to create a list box. A list box lets you select an item from among a list of text items. For example, you might create a list box that permits selection of one file name from a list of file names. If text in the box is too long to be displayed all at once, list boxes have a built-in scroll bar that allows vertical scrolling.

Customizing list boxes

To create a list box whose behavior is different from the standard list box, you can create an *owner-draw list box.* Owner-draw list boxes have only the general characteristics of standard list boxes, so you need to define their functionality in the source code. While standard list boxes draw their own contents, owner-draw list boxes do not. The application takes responsibility for the visual appearance of the list box. For example, you can use an owner-draw list box to display a series of bitmaps rather than text strings. For more information on custom controls, see Microsoft's Software Development Kit.

To define an owner-draw list box:

1. Create a standard list box.
2. Open the list box's style box in either of two ways:
   - Press the *Alt* key and double-click the list box.
   - Select the list box, then choose Attributes from the Controls menu. On the attributes box, click the Style button.
3. Choose either Owner Draw Fixed or Owner Draw Variable:
   - **Fixed**: Items in the list box must all be the same height.
   - **Variable**: Items in the list box vary in height.

Combo boxes

Use the Combo Box tool to create a combo box. Combo boxes combine the features of an edit box and a list box: they let you select an item from among a list of items, but they also let you type a selection into an edit field.

You can create the following styles of combo boxes:

- a simple combo box

- a drop-down combo box
- a drop-down list combo box

To create any style of combo box:

1. Choose the Combo Box tool and draw the box. This creates a simple combo box. If this is the style you want, you're finished. Otherwise:
2. Open the combo box's style box in either of two ways:
   - Press the *Alt* key and double-click the combo box.
   - Select the combo box, then choose *Attributes* from the *Controls* menu. On the attributes box, click the *Style* button.
3. Choose the style of combo box you want.

Simple combo box

A simple combo box always displays both its features: the edit area and the list box. The edit area behaves just like an edit control—it lets you enter and edit text. The text need not match an item from the list in the list box. If it does, the corresponding list item is selected.

As an alternative to typing text into the edit area, you can select an item from the list box. The list box in a combo box behaves just like a standard list box.

Drop-down combo box

A drop-down combo box behaves like a simple combo box. However, when the combo box first opens, its list area isn't displayed. To display the list area, click the down arrow to the right of the editing area.

Drop-down combo boxes are useful when you want to fit a lot of controls into a small area. Initially they take up only as much space as the editing area needs. When the list box is opened, it overlaps the dialog box and doesn't require additional space.

A drop-down combo box is useful when you want to provide a list of selections, but allow an alternative selection. For example, you might use a drop-down combo box in a dialog box for opening or saving disk files. You can use the list box to search disks for an existing file name, or use the edit field to enter the name of a new or existing file.

Drop-down list combo box

A drop-down list combo box is similar to a drop-down combo box. It has a list area that drops down when needed and retracts when not needed. The boxes differ, however, in the behavior of their editing areas. Whereas the editing area in a drop-down combo box lets you enter any text, the editing area of a drop-down list combo box lets you enter only an item from the list box. You can select the item from the list box, or type it into the editing area, but you can't enter an item that isn't listed in the list box.

Drop-down list combo boxes are useful in cases where only the listed selections are acceptable. For example, even though a computer might be linked to several printers, you can only choose one among the several to print to at any given time.

Customizing combo boxes

To create a combo box whose behavior is different from the predefined combo boxes, you can create an *owner-draw combo box*. The advantage of creating owner-draw combo boxes is that they're not limited to displaying text strings. For example, they can display a series of bitmaps.

Owner-draw combo boxes have only the general characteristics of standard combo boxes, so you need to define their functionality in the source code. Whereas predefined combo boxes draw the contents of their list boxes, owner-draw combo boxes do not. The application is responsible for the visual content of the list box portion of the combo box control. For more information on custom controls, see Microsoft's Software Development Kit.

To define an owner-draw combo box, choose either Owner Draw Fixed or Owner Draw Variable on the combo box's Styles dialog:

- **Fixed:** Items in the list box must all be the same height.
- **Variable:** Items in the list box vary in height.

Scroll bars

Use the Scroll Bar tool to create a standalone scroll bar in the dialog box. Edit controls, list boxes, and combo boxes have their own, attached scroll bars and don't need standalone scroll bars.

A standalone scroll bar is useful in a dialog box that needs more than one vertical or horizontal scroll. For example, the graphic editors in the Resource Toolkit use color palettes for drawing bitmaps. You can edit a color in the palette by changing its mixture of red, green, and blue. To let you change the mixture, the editors provide a dialog box with three standalone scroll bars.

The Scroll Bar tool lets you create scroll bars with either standard or adjustable widths.

Standard scroll bar

Use the Standard Scroll Bar tool to create a scroll bar with a standard width. You cannot change the width of a standard scroll bar.

Variable-width scroll bar

Use the Scroll Bar tool to create a scroll bar whose width you can change.

Icons

Use the Icon tool to create a box within which you can place an icon. Windows automatically sizes this box, so all you have to do is position it where you want the icon to appear within the dialog.

Custom controls

Use the Custom Control tool to create a box within which you can place a customized control, giving it characteristics that aren't available with any of the control tools.

A custom control must have a dynamic-link library (DLL) that defines the Windows procedure for the control, and also the functions that interact with the Dialog Box editor. For more information on custom controls, see Microsoft's SDK.

Alignment palette

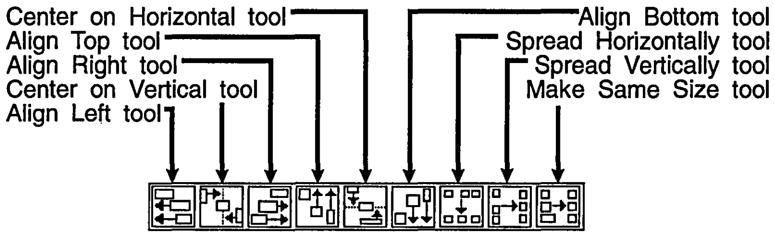
You can align controls within a dialog box using either the Alignment palette or the Alignment menu. Either lets you

- adjust the position of a selected set of controls. The controls can be aligned relative to the top, bottom, left, right, or center of one control from the set. Or they can be spread evenly across the dialog box's height or width.
- adjust the size of a selected set of controls, setting them all equal.

The control used as the standard or anchor for an adjustment is always the first one selected in the set. Thus, if you select two controls and then click the Make Same Size tool, the second control you selected is made the same size as the first one you selected.

Figure 6.2 shows the Alignment palette. Generally, you'll want to use the alignment tools any time you create two or more controls of the same type within the dialog box. That way you can ensure that they're properly aligned and proportioned.

Figure 6.2
The Alignment palette



Alignment tools

The alignment tools are on the Alignment palette. The alignment tools let you select a control to use as an anchor, then select a set of additional controls to arrange in accordance with that anchor. Table 6.1 describes the actions you can do.

To align controls,

- Select the control you want to use as the anchor for the alignment.

Note If you're spreading controls across the dialog box's width or height, this first control isn't used as an anchor. It's simply one of the controls whose position will be adjusted.

- Hold down the *Shift* key and select each additional control you want to align.
- Select an alignment tool from the palette by clicking it with the mouse.

Table 6.1: Alignment tools

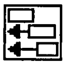
Tool	**Icon**	**Function**
Align left		Aligns all selected controls by the anchor's left border. The vertical position of each individual control is unaffected.

Table 6.1: Alignment tools (continued)

Tool	Icon	Function
Align right		Aligns all selected controls by the anchor's right border. The vertical position of each individual control is unaffected.
Align top		Aligns all selected controls by the anchor's top border. The horizontal position of each individual control is unaffected.
Align bottom		Aligns all selected controls by the anchor's bottom border. The horizontal position of each individual control is unaffected.
Center on horizontal		Aligns the center of all selected controls with the anchor's horizontal axis. The horizontal position of each control is unaffected.
Center on vertical		Aligns the center of all selected controls with the anchor's vertical axis. The vertical position of each control is unaffected.
Spread horizontally		Evenly spreads the selected controls within the dialog box's width.
Spread vertically		Evenly spreads the selected controls within the dialog box's height.
Make same size		Makes all selected controls the same size as the anchor.

Menus

This section outlines the Dialog Box editor's menus, with the exception of the File and Edit menus, which are discussed in Chapter 10.

Dialog menu

The Dialog menu provides the following options:

- **Attributes:** Opens the Attributes box, which lets you define attributes for the dialog box. Defining attributes is discussed on page 56.
- **Show Tab Order:** Displays the order in which focus moves through the dialog box's controls when you press *Tab* in the application. The order is shown by displaying red numbers in the upper-left corner of each control. The numbers start at 1 and progress sequentially. To change the tab order by altering the tab number for a control, use the Controls menu.
- **Show Item ID:** Displays the ID of each control. The ID is displayed in blue in the lower-left corner of each control. To change the ID number, use the control's attributes box, as discussed on page 58.
- **Show Item Group:** Uses patterned rectangles to represent control groups. All controls in the same group show as rectangles of the same pattern. To define logical groups for the controls, use the Controls menu.
- **Grid Items:** Lets you set the grid resolution that determines the placement of controls. For example, setting a grid resolution of 2 means that controls can only be located at even coordinate values within the dialog box.
- **Palettes On Bottom:** Positions the Tools and Alignment palettes below the editing area in the Dialog Box editor.
- **Palettes On Top:** Positions the Tools and Alignment palettes above the editing area in the Dialog Box editor.
- **Swap Palettes:** Switches the locations of the Tools and Alignment palettes.
- **Show Header:** Displays the header-file editor—assuming you opened a header file by choosing File | Open Header from the Dialog Box editor's menu. If the header file editor is currently

displayed, you can hide it with this choice. See Chapter 9 for information on header files and the header file editor.

Controls menu

The Controls menu provides the following options:

- **Attributes:** Opens the currently selected control's attributes dialog box, which lets you define attributes for the control. Defining attributes is discussed on page 56.
- **Set Tabstop:** Turns on the tab stop bit for the currently selected control. This lets the control receive focus when *Tab* is pressed in the application. Set Tabstop is either on or off. A check next to it indicates it's on.

  By default, new controls, except statics, group boxes, icons, and custom controls, are defined as tab stops.

  To remove tab stop status from a control currently defined as a tab stop, select the control, then choose Set Tabstop from the Controls menu to turn off the tab stop status.

For group boxes, Start Group is always shown as checked and disabled.

- **Start Group:** Defines controls as members of a logical group. A logical group defines a set of controls within which input focus can be moved by pressing the arrow keys.

  To define controls as a logical group, select the control you want as the first member of a new group, then choose Start Group from the Controls menu. This turns Start Group on for the selected control. Any new controls placed in the dialog box after selecting this become part of this group.

  For groups that are already defined, you can change group membership by using Group Together on the Controls menu.

  Start Group is either on or off for a control. If Start Group is on for a control and you want to turn it off—perhaps so you can change the control's group—select the control, open the Controls menu, and turn off the Start Group option. A check by the menu option indicates it's on.
- **Move Forward:** Moves the selected control forward one tab stop. For example, if you select a control with a tab order value of 6, and then choose Move Forward, the control moves to position 5 in the tab order. The control previously in position 5 moves to position 6. The rest of the tab order isn't affected.
- **Move Backward:** Moves the selected control back one tab stop. For example, if you select a control with a tab order value of 3, and then choose Move Backward, the control moves to position

4 in the tab order. The control previously in position 4 moves to position 3. The rest of the tab order isn't affected.

- **To Bottom of Tab Order:** Moves the selected control to the last tab stop in the tab order. All controls in the tab order after the selected control move up one position.
- **To Top of Tab Order:** Moves the selected control to the first tab stop in the tab order. All controls in the tab order before the selected control move back one position.
- **Group Together:** Creates a new group. To define control groups, select the controls you want to group together, then choose Group Together from the Controls menu. This associates all the selected controls into a group and moves them to the top of the tab order.

  This operation doesn't work if multiple controls from the group you select in the first step have Start Group turned on.

Tools menu

You can use the Tools menu as an alternative to using the tools on the Tools palette. The sections "Dialog Box tool" and "Control tools" on pages 60 and 61, respectively, discuss each tool.

Alignment menu

You can use the Alignment menu as an alternative to using the tools on the Alignment palette. "Alignment tools" on page 68 discusses each alignment tool.

C H A P T E R

7

The Menu editor

The Menu editor lets you create and edit menu resources. This means you can determine both the content and appearance of any menu in a Windows application. In addition, you can interactively test changes you make to a menu without having to compile and run the application.

Files

The Menu editor can edit menu resources from resource (.RES) files, executable (.EXE) files, and dynamic link libraries (DLLs). It can save a menu resource directly into a new or existing .RES file, or into an existing .EXE file. In addition, it can save the same menu specification into a resource script (.RC) file. For more information on these files, see Chapter 2.

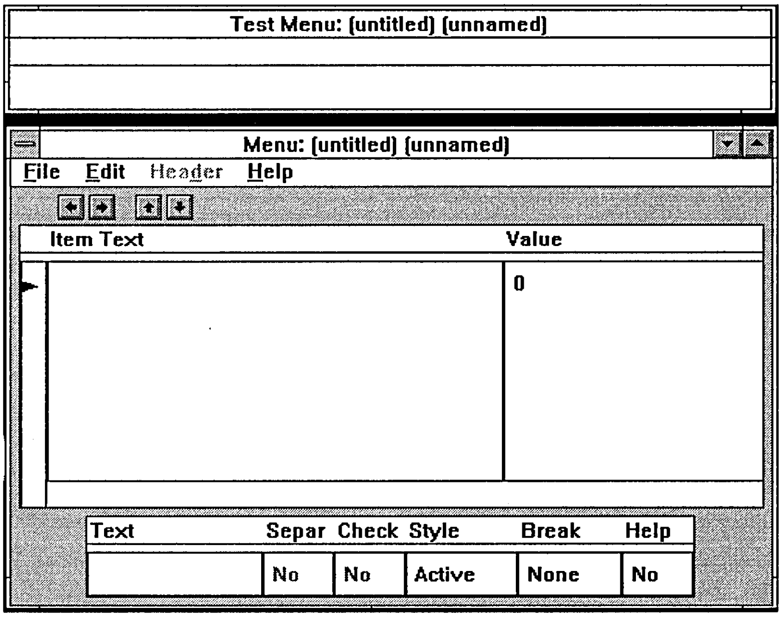
The Menu editor can also open a header file (also called an include file) to correlate symbolic constants from the file with the menu resource. This adds a Symbol column to the menu table, and opens the Header editor. You can edit symbol values with the Header editor, or edit them directly from the menu table. For more information on header files, and for instructions on editing symbol values, see Chapter 9.

Using the editor

If you need instructions for creating or opening a file, see page 23.

The Menu editor has several features:

- A menu table
- Movement buttons
- Style and attribute fields
- A test window



Text	Separ	Check	Style	Break	Help
	No	No	Active	None	No

Figure 7.1
The Menu editor

Menu table

When you first open the Menu editor, it contains a menu table with two columns of edit fields. The Text column lets you enter or edit menu text, which is the text displayed for menu items. The Value column lets you enter or edit a value for the current item. If a header file is open, an additional column shows the symbolic name, if any, for each menu item. The edit fields in the menu table are described on page 76.

Movement buttons Above the menu table are four movement buttons. These buttons have arrows pointing up, down, left, and right on them. The Left and Right movement buttons let you change a menu item's level in the menu hierarchy. The Up and Down movement buttons let you change a menu item's row in the Text field, and therefore its position on a menu. See "Defining menu levels" (page 77) and "Changing the position of menu items" (page 78) for more information on movement buttons.

Style and attribute fields Below the menu table is a set of fields that let you define menu styles and attributes. See "Defining menu styles and attributes" on page 81 for more on style and attribute fields.

Test window Above the menu table is a special test window. The test window lets you interactively test your menu. See "Testing menus" on page 84 for more about the test window.

Defining menu text

In the menu table, the fields you can edit are the Text and Value fields (and when a header file is open, the Symbol field). Use the editing conventions discussed in Chapter 10 to enter and edit text in these fields.

Navigating the editor

Navigate the menu table with the keys discussed in Chapter 10. In addition,

- To add a new row to the table, press *Ctrl-Enter.* This works regardless of which field is active when you press *Ctrl-Enter.* Default values are automatically entered for those columns that use defaults.
- To move the active field between the menu table and the style and attribute fields, press *Ctrl-Tab.*

Editing text

You can edit the Item Text, Value and, when a header file is open, Symbol fields. The remaining fields are selection fields. Edit these fields as discussed in Chapter 10.

- **Item Text field.** The text in this field represents the menu text that appears in the actual menu. Enter the text exactly as you want it to be displayed on the menu. The text can contain any alphanumeric characters, and can be up to 255 characters long. Windows uses the **&** character to define and underscore activation keys, and **\t** for a tab. These characters are discussed in more detail on pages 80 and 80.
- **Value field.** This field contains the ID number for the menu item specified in the Text field. This is the number the item returns when it's selected, and it's the number used to identify the item in the application's source code. As an alternative to using the value, you can define a symbol for the menu item in the Symbol column.

  The value assigned to a menu item must be an integer between 0 and 65,535. The value must be the same number used to identify the item in the source code. When a new row is created, its value is set to 0.

  The ID number for each menu item within a window class must be unique. However, the same IDs can be used for menu items within another window class. For example, consider the Dialog Box editor and the String editor. The IDs for all menu items used in the Dialog Box editor must be different from each other. However, they can be the same IDs as menu items used in the String editor. This is because the menu items are defined in different window classes. For any window class, the ID values of menu items can be the same as the IDs used in other resources, such as for the controls in a dialog box. This is because Windows interprets the ID value within the context of the resource currently being used in the application.

  A menu item with a pop-up menu under it opens the pop-up menu when selected; it doesn't return a value. Thus, you don't have to assign a value to an item if you plan to create a pop-up menu under it. Once you create a pop-up menu under an item, the item's Value field is grayed out. If you assign a value to an item before the pop-up is created, the value is grayed out when the pop-up is created. This lets you create a pop-up menu under an item that previously returned a value. If you later delete the pop-up menu, the item's value becomes active again.

- **Symbol field.** This field contains the symbol whose ID number in the header file matches the value for the current menu item. You can use this symbol in the application's source code to refer to the menu item.

Defining menu levels

Menus in Windows applications are hierarchic. Top-level menu items are displayed on a menu bar, which is just below a window's title bar. In most cases, when you choose an option from the menu bar, a pop-up menu opens to display a list of options. For example, when you choose File from the menu bar, a pop-up menu offers choices such as New for creating a new file, and Open for opening an existing file. There can be any number of pop-up levels under an item on the menu bar.

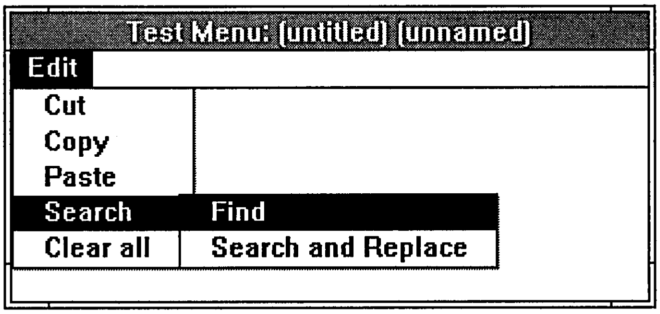
*You **must** use the movement buttons or the Alt-arrow method to change a menu item's level in the hierarchy.*

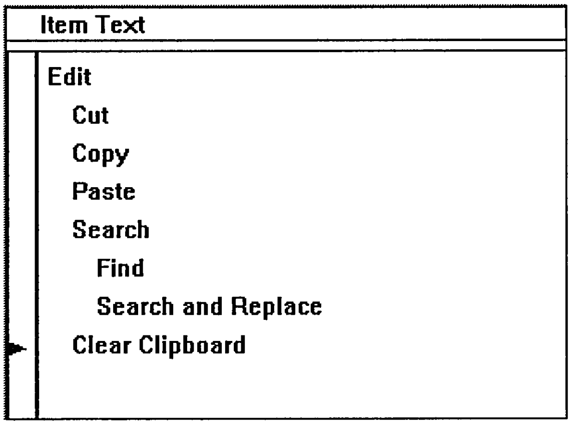
To represent pop-up levels, the Menu editor indents the items on the pop-up. Thus, to create a new pop-up level, indent text using the Right movement button.

To move an item up or down in the menu hierarchy, activate any of the item's fields. Click the Left movement button to move it up in the hierarchy, or the Right movement button to move it down. Alternatively, hold down the *Alt* key and press ← or → on your keyboard. Either method shifts text for the current item left or right one place.

If the item has pop-up menus under it, all the pop-up menus are moved with it. Figure 7.2 shows three levels of text in the Text field. The Test Window in the figure shows that the indented text results in three menu levels.

Figure 7.2
Defining menu levels





Changing the position of menu items

To change the position of menu items, use the Up and Down movement buttons.

To move an item up or down in the menu order, activate any of the item's fields. Click the Up movement button to move it up in the order (it has a solid arrow pointing up), or the Down movement button (a solid arrow pointing down) to move it down. Alternatively, hold down the *Alt* key and press ↑ or ↓ on your keyboard. Either method moves text for the current item up or down one place.

An item can be moved only to a position at its current level in the menu hierarchy. If there isn't a position available at its current level, the item isn't moved. If the item has pop-up menus under it, all the pop-up menus are moved with it. For example, consider the following outline structure for the Text field:

```
File
  New
  Open
Help
```

- If you activate Open, then click the Up movement button, Open is positioned above New.
- If you activate Open, then click the Down movement button, Open is moved below Help. This effectively moves an item from one menu to another.
- If you activate Help and click the Up movement button, Help is positioned above File. Conversely, if you activate File and click the Down movement button, File and its pop-up items New and Open are positioned after Help.

Setting an activation key

The easiest way to make menu selections is to use the mouse. Nonetheless, any good user interface provides easy keyboard alternatives to using the mouse for making menu choices. This can be done in two ways: by defining an accelerator key, which automatically invokes the command no matter where you are, and by defining an activation key, which is the key (usually a letter) that you press when a menu is open to invoke a command from that menu. Here is how to do each:

1. Define an accelerator key for the menu item. Generally, the number of accelerators defined for an application is small compared to the number of functions available through the menus. For example, many Windows applications define accelerators for the Cut, Copy, Clear, Insert, and Paste functions on an Edit menu, while they don't define accelerators for many other menu items in the application.

   To define accelerator keys, use the Accelerator editor (discussed in Chapter 4). You can't *define* an accelerator in the Menu editor. However, you should *identify* a defined accelerator in the menu text, usually separating it from the rest of the menu text with a tab, as discussed in the next section.

Windows automatically defines the Alt key as the activation key for the menu bar.

2. Define an *activation key* for the item. An activation key is a key used to choose a menu item. For example, if *F* is the activation key for File and *S* is the activation key for Save, you can

choose File | Save by pressing *Alt* to activate the menu bar, *F* to choose the File menu, and then *S* to choose Save. The activation keys for each menu work only when the menu is active.

You should define an activation key for every menu item. Usually the best choice for the activation key is the first letter of the menu text. In cases where the first letter isn't unique on the current menu, you can define any suitable alternative. For example, most File menus provide the options Save and Save As. The *S* key is usually defined as the activation key for Save, and the *A* key is usually defined as the activation key for Save As.

The same key shouldn't be used to activate more than one item on the same menu. If it is, you may have to cycle through those items to select the one you want. However, you can use the same key on different menus. For example, *P* can be used to activate Print on the File menu, and Paste on the Edit menu. If the File menu is active, the *P* can only select Print. It can't select Paste unless the Edit menu is active.

Defining an activation key

The & doesn't have to precede the first letter in a word. For example, Cu&t defines T as the activation key for Cut.

To define an activation key, type an ampersand (&) in front of the corresponding letter in the menu text. For example, the text *&Save* defines *S* as the activation key for Save. The text *Save &As* defines *A* as the activation key for Save As.

In the actual menu, the ampersand is translated to an underscore under the activation key. In the menu table, the ampersands in the menu text define one activation key for each menu item. The Test Window shows that the activation key for each item is identified by an underscore in the actual menu.

Note By default, Windows uses an item's first letter as its activation key. However, Windows doesn't identify the default activation key with an underscore, and doesn't resolve the conflict when two items on a menu have the same first letter. For these reasons, it's usually best to define the activation keys in the Menu editor.

Inserting tabs in menu text

You can insert a tab in menu text by typing \t where you want the tab to go in the text. For example, in the text Clear\tDel, the tab will be inserted before the string Del.

Tabs are most useful when you're identifying accelerator keys for a menu option. The tab ensures that the text identifying accelerator keys is aligned in a column. Windows automatically determines the appropriate width and spacing for the column based on the menu text.

Defining menu styles and attributes

The Menu editor lets you define different styles and attributes for menus. You can

- add separator lines to a pop-up menu
- use a checkmark to indicate an item's status
- define a style for a menu item's text
- align menu items in columns
- assign the help attribute to a menu item

The following sections discuss these attributes and styles in detail.

Note In Windows applications, the menu features owner-draw menu items, checkmark bitmaps, and bitmaps aren't implemented through a menu resource. Rather, they're dynamically added to menus through Windows function calls. For that reason, they can't be created or edited in the Menu editor.

Adding separator lines to pop-up menus

A *separator* is a horizontal line that divides menu items into visual categories. Separator lines are optional on a menu and serve no function beyond providing visual categories for the menu items.

Use separator lines to separate menu items into functional categories. To add separator lines to a menu,

1. Insert a blank row in the menu table where you want the separator line to appear. To do so,
   - Determine which item will immediately precede the separator line, then activate any of its fields.
   - Press *Ctrl-Enter.* This inserts a blank row beneath the current item.
2. Type `Yes` in the Separator field. This defines the row as a separator line. All the other style and attribute fields for the row become inactive.

Using a checkmark

Items on a pop-up menu can provide independent options, each of which can be turned on or off. They can also provide a set of mutually exclusive options, only one of which can be on at any given time. In these cases, you can use a checkmark to indicate whether an option is in effect. To use a checkmark for a menu item, type `Yes` in its Check field.

Defining style for menu items

Some menu items can't be used until a certain condition is met. For example, the Cut and Copy options on an Edit menu can't be used until you select something to cut or copy. The Paste option can't be used unless the Clipboard contains something to paste.

For these types of menu items, the Menu editor lets you use different styles of text to reflect the current status. For example, you can display Cut, Copy, and Paste as gray text when these options are unavailable.

To specify a menu item's style, type `Active`, `Inactive`, or `Gray` in its Style field:

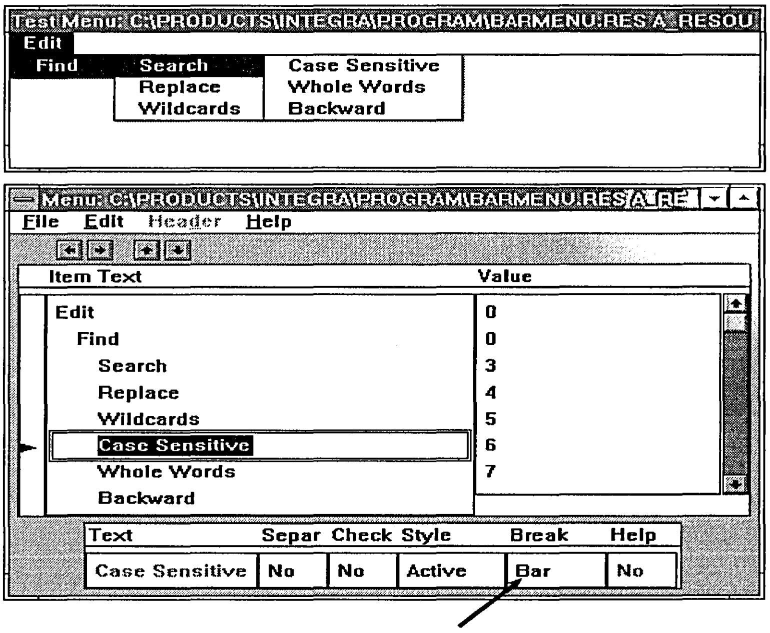
- **Active** indicates the item is active and returns its defined value when selected. An active item is displayed in the normal style.
- **Inactive** indicates the item is initially inactive and doesn't return its defined value when selected. Inactive items are displayed in the normal style; they can be useful as you develop your application. For example, if you don't have code written for an item's function, but you still want to display it as it will appear in the final application, you can define the item as inactive.
- **Gray** indicates the item is to be shown as gray text initially and therefore won't return a value when selected. Gray text flags items as currently unavailable.

The source code defines the conditions under which an item is active, inactive, or grayed. When conditions are met in the application for an inactive or grayed menu item to become available, the source code changes the item's style to active.

Aligning menu items in columns

To align menu items in columns, you can define breaks for the items. Windows lets you define two types of breaks: a column break and a bar break. A *column break* aligns items in a column. A *bar break* aligns items in a column, and also separates columns with a vertical bar.

Use breaks when you have too many items to fit in one column, but you don't want to use pop-ups for some of the items. You can also use breaks to encourage comparisons between the items listed in each column. The following figure shows a menu with both column breaks and bar breaks:



Test Menu: C:\PRODUCTS\INTEGRA\PROGRAM\BARMENU.RES A_RESOU
Edit
Find
Search
Replace
Wildcards
Case Sensitive
Whole Words
Backward
Menu: C:\PRODUCTS\INTEGRA\PROGRAM\BARMENU.RES A_RE
File
Edit
Header
Help

Item Text	Value
Edit	0
Find	0
Search	3
Replace	4
Wildcards	5
Case Sensitive	6
Whole Words	7
Backward	

Text	Separ	Check	Style	Break	Help
Case Sensitive	No	No	Active	Bar	No

Figure 7.3 Using breaks

To define breaks on a pop-up menu, determine which item should be the first item in a new column, then type `None`, `Column`, or `Bar` in its Break field:

- **None** indicates the current item doesn't begin a new column.
- **Column** indicates the current item begins a new column. All text below this item in the Text field is aligned in the new column, unless you set another break.

- **Bar** works the same as Column, except that it inserts a vertical bar between columns to further emphasize the break.

Items below the break item on the menu are aligned in its column, unless you define another break.

Assigning the help attribute

The Menu editor lets you assign the Windows *help attribute* to a top-level item on a menu. The help attribute determines that the item will be right-justified on the menu bar. This is useful when you want to separate the Help menu from other menus on the menu bar.

Only one item can have the help attribute.

To assign the help attribute to a menu item, type `Yes` in its Help field. Items positioned below the help item on the menu are also right-justified on the menu bar. To avoid this, make the help item the last top-level item in the menu.

Testing menus

The Test window lets you view the results of your work. Use the menus in the Test window exactly as you would use them in the actual application. When you make a selection in the Test window, its defined value and symbol are displayed in the window's client area.

The Test window is automatically refreshed each time you click it. Thus, every time you use it, it reflects the current status of your work.

C H A P T E R

8

The String editor

The String editor lets you create and edit string resources. String resources are strings used by an application for any of the text it displays, including

- error messages
- status messages or other general system messages
- caption text in a window

The string resources are loaded as needed from an executable file. The more strings you define as resources, the easier it will be to translate displayed text to another language, or customize the text for different uses.

Defining strings

If you need instructions for creating or opening a file, see page 23 in Chapter 3.

The String editor stores strings in a *string table* as follows:

- The first column in the table shows the string ID, an integer.
- The second column shows the string itself.
- If a header file is open, a third column shows the symbolic name of the ID.

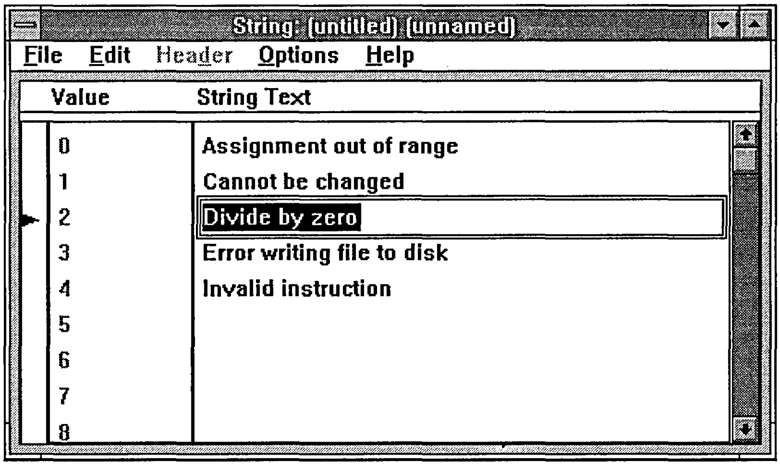
The maximum size of a string table permitted in Windows is 64K. You can define only one string table for an executable file.

Field and row indicators

The String editor uses a *field indicator* to show which field in the table is active, and a *row indicator* to show which row contains the active field. The field indicator is a rectangular frame surrounding the active field, and the row indicator is an arrow head in the table's left margin.

Figure 8.1
The string table

The indicators show that the active field is the String Text field for Value 2.



Value	String Text
0	Assignment out of range
1	Cannot be changed
2	Divide by zero
3	Error writing file to disk
4	Invalid instruction
5	
6	
7	
8	

The string table

The first column in the string table is the Value field. This field holds the ID number for the string in the String Text column (column two). To load the string text into your application, refer to this number in the application's source code. Alternatively, you can define a symbol for the string, as discussed on page 88.

The String editor automatically assigns values to the rows in a string table; the values are sequential integers, beginning with 0. Rows cannot be added or deleted from the table, nor can a row value be changed.

The ID values of string text can be the same as the IDs used in other resources, such as for the controls in a dialog box. This is because Windows interprets the ID value within the context of the resource currently being used in the application.

String text field

The second column, the String Text field, shows the string itself. Enter the strings without quotation marks; if a string is enclosed in quotes, the quotes are treated as part of the string. String text can be up to 255 characters long.

When the string text for any value is loaded into an application, the string is displayed on a single line. To add carriage returns within the string, use the code \012 to indicate the position of the carriage return. For example, the string text

Line one\012Line two

is displayed as

```
Line one
Line two
```

in the application.

Symbol field

If a header file is open, a third column, called Symbol, shows the symbol whose ID number in the header file matches the value for the current string text. To load the string text into your application, you can refer to this symbol in the application's source code. This appears if a header file is open; see page 88 for more on header files.

String sizes and values

➡ Windows loads string resources in 16-string segments. For example, when you ask for string 14, Windows loads strings 0 through 15. To reduce the number of loads Windows has to perform, we recommend that you group strings that are common to a particular function into 16-string segments. This allows Windows to load related strings together, and also discard them together when the application no longer needs them.

Except for grouping functionally related strings into 16-string segments, there is no significance to the value you choose for defining new string text.

When the strings are saved, only those rows containing data are written to the resource file. However, when the resource file is subsequently opened in the String editor, the blank rows are inserted again so you can add additional strings to the table.

When you save a new string table, the String editor opens the Resource Attributes dialog box. Whenever possible, make the table moveable and discardable. For more information on the Resource Attributes dialog box, see "File menu" in Chapter 10.

Navigating the editor

See page 98 for a description of most of the key commands needed to move around in the String editor. In addition, you can page through screens in the string table without skipping any screens by using *PgUp* and *PgDn*. To page while omitting screens that don't have defined string text, use *Ctrl-PgUp* and *Ctrl-PgDn*. For example, if the current screen displays rows 0 through 8 and row 999 is the next row to have defined string text, press *Ctrl-PgDn* to get to row 999.

To go directly to a line, use *Ctrl-L*. For example, to go directly to line 999, press *Ctrl-L*, then type `999` in the resulting dialog box's entry field.

Because you can't add rows to a string table, both *Enter* and *Ctrl-Enter* advance the cursor to the next row of the table. Neither key inserts a blank row or splits string text.

Editing text

You can edit String Text fields and, when a header file is open, Symbol fields. If you select a field from either of these columns, all text in the field is selected.

- To replace the text, begin typing or press *Del*.
- To preserve the text, move the cursor in the field using the mouse, an arrow key, or the *Home* or *End* keys. You can then edit some portion of the text.

Header files

To open a header file so you can browse or edit symbols while defining string resources, choose File | Open Header from the String editor's menu. This adds a Symbol column to the string table.

Opening a header file opens a special header file editor. You can edit symbol values with the header file editor, or edit them directly from the string table. For more information on header files and instructions on editing symbol values, see Chapter 9.

C H A P T E R

9

The Header editor

The Header editor lets you create and edit symbolic constants, called *symbols*, that can substitute for the ID numbers Windows uses to identify resources. Once the symbols are defined, you can load resources from your source code by referring to their defined symbols rather than their numeric values. The Header editor lets you define the symbols while you're creating or editing the resources they substitute for.

Header files

Symbolic constants saved by the Header editor are stored in header files, which are used to include the symbolic constants in the source code. Because header files include constants in the source code, they're also called include files.

Symbols from the header file become *global constants* when they're compiled. Each symbol must therefore be unique within the application.

Here's part of a typical header file with C-style constants:

```
#define CW_FILE_NEW  101
#define CW_FILE_OPEN 102
#define CW_FILE_SAVE 103
```

The file uses **#define** statements to define the symbols as substitutes for their corresponding values. For example, CW_FILE_NEW is defined as the symbol for value 101.

You don't have to use the Header editor in order to create or edit header files. They're text files that can be edited separately with any text editor. When editing a header file with a text editor, you can include comments in the file. For example, you can use C-style comments beginning with /* and ending with */ in the file. These comments are ignored when the header file is compiled.

Header files are included in the application's resource script (.RC) file. In the .RC file, a header file must precede the resources whose symbols it defines. This is necessary because the resources are referred to by their symbolic constants, and the constants must be defined in the source code before they can be used.

Starting the editor

Because the Header editor lets you define symbols while you create or edit the resources they substitute for, never start the Header editor by itself. Rather, begin editing the resource, then start the Header editor from within the resource's editor. For example, to define symbols for string resources, first start the String editor, then, from within the String editor, start the Header editor.

The Accelerator, Dialog Box, Menu, and String editors each have options on their File menu for creating a new header file or opening an existing header file. Once you choose one of these options, the Header editor is automatically started, and the header file is opened. Only one header file at a time can be open within each resource editor.

Note The Header editor is a child window of the editor used to start it. Thus, if you minimize or close the associated editor to an icon, the Header editor is also minimized or closed.

Creating new header files

To create a new header file, choose New Header from the File menu in the Accelerator, Dialog box, Menu, or String editor. This opens an empty header file so you can define new symbols for your application.

Typically, you would create only one new header file for each application you develop. This is because it's easier to ensure the symbols are unique within a single header file than it would be to ensure they're unique within multiple header files.

However, determining the number of header files to create for an application is a design decision. You can create as many header files as you need. For example, you can create a separate header file for Accelerator, Dialog box, Menu, and String resource types. To ensure that symbols in each header file are unique, you can prefix them with the first letter of the resource type. For example, you might prefix *A_* on each accelerator symbol.

Opening existing header files

To open an existing header file, choose Open Header from the File menu in the Accelerator, Dialog box, Menu, or String editor. If you need more instructions for opening a file, see the instructions for the File menu in Chapter 10.

Using the editor

The Header editor lets you edit or delete symbols, move symbols in the header file, and save changes you make to the header file.

ID numbers

Before working with a header file, determine whether you want the resource ID numbers (values) to be displayed in decimal format or hexadecimal format. The radio buttons in the top right corner of the Header editor let you select the appropriate format. Decimal format is the default.

A defined symbol becomes a global constant in the application source code. Thus, if you include more than one header file in your application, the symbols in each file must be unique. No symbol can be defined in more than one file, unless it's associated with the same value. For example, if symbol CW_FILE_NEW is associated with value 101 in one header file, no other header file created for the same application can contain symbol CW_FILE_NEW, unless it's also associated with value 101.

Values don't have to be unique within the header file because different resources in an application can use the same ID values. For example, a menu item on a menu might have value 101, and a button on a dialog box might also have value 101. Windows won't confuse the two values because it interprets the values within the context of the resource being used.

The Header editor searches for values from the top of the header file to the bottom, and displays the first symbol it finds for a value. This means the order of symbols in the header file can determine which symbol is displayed for a value in the associated editor. For example, if you define symbol CW_FILE_NEW for menu item 101 in the Menu editor, and SAV_BUTTON is defined for value 101 higher in the header file, SAV_BUTTON is displayed in the Menu editor as the symbol for menu item 101. To display CW_FILE_NEW as the symbol, you must move it above SAV_BUTTON in the header file.

Editing a symbol

You can create or edit a symbol in the Header editor, or in the associated resource editor. Each subsection in this section gives separate instructions for the different editors.

In the Header editor

To edit a symbol in the Header editor,

1. Make the symbol field active and type the symbol into the field. For an existing symbol, you can select the symbol from the list box, and it is automatically copied into the Symbol field. Once you type the symbol, make the Value field active and type the resource's value:

   - **Symbol.** The first character of a symbol must be alphabetic. All additional characters can be alphanumeric. The maximum length of the symbol depends on the source code language.

     The symbol must be unique. If you enter the name of an existing symbol, the value you specify in the Value field replaces the value previously defined for the symbol—even if the existing symbol isn't currently visible in the list box.
   - **Values.** Enter the value for the resource the symbol will substitute for. For example, if you're defining a symbol for a menu item with ID 101, type `101` as the value to associate with the symbol. The value must be numeric. To enter a long number, suffix the number with an *L*.

You cannot enter C-style macros or expressions in the Value field. The Header editor ignores any lines it doesn't understand.

2. Click To Top, Put, or To End to position the symbol in the header file:

   - **To Top** places the symbol at the top of the header file. This ensures that the symbol will be displayed in the associated resource editor, even if another symbol in the header file is defined for the same value.
   - **Put** places the symbol into the row currently selected in the list box. You might choose Put if you just edited a symbol in the table and don't want to change its position in the header file. Or, you might choose Put to put a new symbol into the selected row.
   - **To End** places the symbol at the bottom of the header file. This ensures that the symbol won't be displayed in the associated resource editor if another symbol is defined for the same value. You might do this to test whether the value is unique in the header file, or to prevent the new symbol from affecting your work in the associated resource editor.

     For example, if you just defined symbol CW_FILE_NEW for menu item 101 in the Menu editor, you could place the symbol at the bottom of the header file, then see whether CW_FILE_NEW is the symbol displayed for menu item 101 in the Menu editor. If it is, you know value 101 is unique in the header file. If it isn't, you know there's at least one other symbol defined for value 101 in the file.

     Or, you might be working in the String editor when you decide to define symbol CW_FILE_NEW for menu item 101. If you also have a defined symbol for a string with value 101, you want to be sure the symbol for string 101 is above the symbol for menu item 101 in the header file. Otherwise, the String editor displays the wrong symbol for the string resource that has value 101.

In the Accelerator, String, and Menu editors

To edit a symbol in the Accelerator, String, or Menu editor,

1. Select the Symbol field from the appropriate row in the table, then add or edit the symbol.

2. Press *Enter,* or use the mouse or keyboard keys to move the field indicator to another field. This opens the Undefined Symbol dialog box.
3. Type the value you want the symbol to substitute for. This will most likely be the value for the current Symbol field, but the dialog box lets you change your mind and associate the symbol with a different value.
4. Click Put Top or Put End to position the symbol in the header file. Put Top and Put End are equivalent to To Top and To End in the header file.

For a discussion of the Symbol and Value fields, and the To Top and To End push buttons, see the instructions on page 92 for editing the symbol in the Header editor.

In the Dialog Box editor

To edit a symbol in the Dialog Box editor,

You can't enter alphabetic characters in the Item ID field unless a header file is open.

1. Open the control's Attributes box by double-clicking the control. In the Attributes box, the Item ID field displays the control's value. If there's a symbol defined for that value in the header file, the symbol is displayed instead.
2. Type the new symbol into the Item ID field, then click the OK button. This opens the Undefined Symbol dialog box.
3. Type the value you want the symbol to substitute for, then click Put Top or Put End to position the symbol in the header file. Put Top and Put End are equivalent to To Top and To End in the header file.

For a discussion of symbols and values, and the To Top and To End push buttons, see the instructions on page 92 for editing the symbol in the Header editor.

Deleting a symbol

To delete a symbol, select it in the Header editor's list box, then click the Delete button. This deletes both the symbol and its associated value from the header file. Be sure to select the correct symbol. *The Header editor does* not *open a dialog box to ask whether you want to delete it.*

Moving a symbol

To move a symbol within the header file, select it in the Header editor's list box, then click To Top, Put, or To End. For a discussion of these buttons, see page 92.

Saving the header file

To save the header file, click Save or Save As:

- **Save** saves the file under its original name.
- **Save As** saves the file under a new name.

To make it easier to associate the header file with your application, assign the same file name to the header file as you assign to the application's resource (.RES) file or executable (.EXE) file.

Header files are written in a format compatible with C, C++, and Microsoft's Resource Compiler. The Header editor automatically appends the file extension .H to these files.

If you need more instructions for saving a file, see the instructions for the File menu in Chapter 10.

C H A P T E R

10

Common keys and menus

Each of the Resource Toolkit's editors is documented in a separate chapter in this manual. However, a number of the editors have common functions:

- The Accelerator, String, and Menu editors use tables for entering text and values. Most of the keys you use to navigate the tables work the same in all these editors. Exceptions or additional keys are noted in the specific editor's chapter.
- All the editors have the same options for their File, Edit, and Header menus.

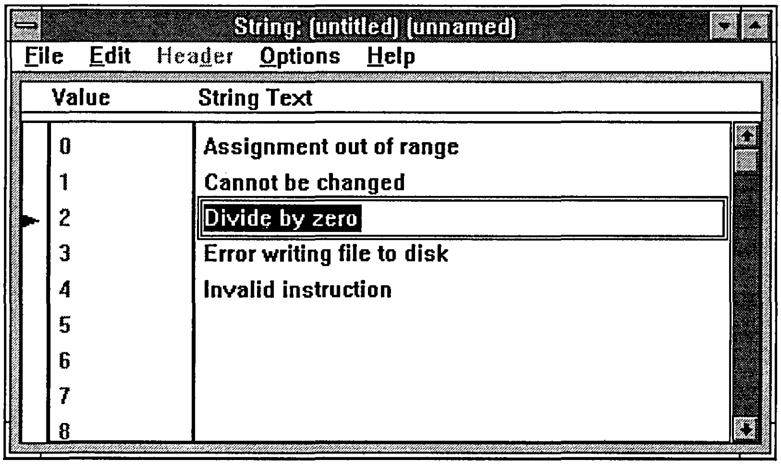
Because these functions are shared by so many of the editors, they're documented together in this chapter.

Editing a table

Tables in the Accelerator, String, and Menu editors have fields you can edit and fields that let you select one of several available options. You use these fields the same way in all the editors.

Field and row indicators. The tables use a *field indicator* to show which field in the table is active, and a *row indicator* to show which row contains the active field. The field indicator is a rectangular frame surrounding the active field, and the row indicator is an arrow head in the table's left margin. The following figure shows a string table. In the table, the indicators show that the active field is the String Text field for Value 2.

Figure 10.1
String table in the String editor



Value	String Text
0	Assignment out of range
1	Cannot be changed
2	Divide by zero
3	Error writing file to disk
4	Invalid instruction
5	
6	
7	
8	

Navigating a table

You navigate a table much the same way you'd navigate a spread-sheet or database, using the following keys:

Key	Action
↑	Moves the active field up one row.
↓	Moves the active field down one row.
Tab	Moves one column to the right.
Shift-Tab	Moves one column to the left.
→	Moves the cursor one position to the right in the active field. In some tables, if the active field isn't an edit field, the right arrow key moves the active field one column to the right.
←	Moves the cursor one position to the left in the active field. In some tables, if the active field isn't an edit field, the left arrow key moves the active field one column to the left.
Home	Moves the cursor to the beginning of the active field.
End	Moves the cursor to the end of the active field.

Typing in an edit field

If you activate a field from any column of edit fields, all text in the field is selected.

- To replace the text, begin typing or press *Del.*
- To preserve the text, move the cursor in the field (using the mouse, an arrow key, or the *Home* or *End* key). You can then edit some portion of the text.

In addition, the tables use standard Windows keys for selecting text, as shown in the following table:

Key	Action
Shift-→	Selects the character one position to the right.
Shift-←	Selects the character one position to the left.
Shift-End	Selects all characters from the current cursor position to the end of the line.
Shift-Home	Selects all characters from the current cursor position to the beginning of the line.

Selecting options in a selection field

Some fields in the tables let you select one option from a set of available options. In all of these fields, use the same method to select an option:

- To cycle through available selections in the field, press *Spacebar.*
- Alternatively, type the first letter of the selection. For example, in a Yes/No field, if Yes is the current value, press the *N* key to cycle to No.

Using menus

The easiest way to make menu selections is to use your mouse. However, the Resource Toolkit observes the standard conventions for using the keyboard to operate menus:

- When the menu bar is active, open the Control-box menu by pressing *Spacebar.*
- Activate the menu bar by pressing *Alt* or *F10.*

- Choose a menu option by pressing the letter that's underscored for it, or use the arrow keys to select the option, then press *Enter.*
- Cancel any menu selection by pressing *Esc.*
- Toggle between the last two windows used by pressing *Alt-Tab.* This action won't toggle to minimized windows.
- Toggle between all open windows by pressing *Alt,* keeping it pressed, and repeatedly pressing *Tab.* When you reach the window you want to work in, release the *Alt* key.

Many menu options can also be implemented with accelerator keys (hot keys). Where they apply, accelerators are listed to the right of their corresponding option on a menu.

The File menu

The File menu in each editor lets you create new files, work with existing files, or define resource attributes.

The File menu in the Accelerator, Dialog Box, Menu, and String editors also lets you open a header file so you can correlate constants defined in the header file with the resource in your current editor. The Bitmap, Cursor, and Icon editors don't use header files and so don't provide this option.

New

New clears the editor and opens a new, untitled file. Use it to clear the current resource out of the editor and enter a new resource.

- If the resource in the editor has been saved, the new file is opened immediately.
- If the resource in the editor hasn't been saved, a dialog box opens to ask whether you want to save changes made to the file:
  - Click Yes to save the changes. This opens a dialog box so you can name the file. If you need instructions for using the dialog box, see the section in Chapter 3 for creating a new resource file.
  - Click No to discard the changes.

Open

Open clears the editor and loads an existing file into it. Use it to clear the current resource out of the editor and retrieve contents from an existing resource.

See Chapter 3 for instructions on opening an existing file and choosing an individual resource to edit.

- If the resource in the editor has been saved, the existing file is opened immediately.
- If the resource in the editor hasn't been saved, a dialog opens to ask whether you want to save changes made to the file:
  - Click Yes to save the changes. This opens a dialog so you can name the file. If you need instructions for using the dialog, see Save As.
  - Click No to discard the changes.

The procedure for opening a file with File | Open is the same as using the Open button in the Resource Manager to open it. Once the file is opened, however, the resource type you can edit is automatically selected for you by the current editor—you can't choose among all the resource types stored in the file.

Save

Save saves the current resource under its original name. Use it to save the resource, then continue working with it. If the resource wasn't originally from a file but was created in a new file, Save works the same as Save As.

Save As

Save As saves the current resource under a different name. Use it when

- you've created a new resource and want to save it.
- you've opened and modified an existing resource and want to save the modified resource without changing the original resource.

You can use Save As to create these kinds of files: .BMP (Bitmap editor), .CUR (Cursor editor), .DLG (Dialog Box editor), .ICO (Icon editor), .RC (Accelerator, Menu, and String editors), and .RES (all editors). You can save into a new or existing .RES file. If you save into an existing .RES file, Save As functions the same as Save Into.

When you click Save As, the File Save dialog box opens so you can name the resource. Available files are listed in the dialog box's files box, with the current directory identified above the box. The Directories box shows available directories.

To store the file in a different directory, change the directory. You can either create a new file or replace one of the existing files that are displayed in the Files box.

To create a new file in the Files box,

1. Set the type of file you want to create using the File Type radio buttons. These radio buttons are to the right of the selection boxes. For a definition of each of the file types, see Chapter 1.
2. Position the cursor in the entry field and type a name for the file. Don't include a file extension; the extension is taken from the File Type radio button.

To replace a resource of the same name within one of the existing files listed in the Files box,

- Select the file. This automatically copies the file's name into the entry field, and also selects the corresponding File Type radio button.

When you're through, click the dialog box's Save button.

- If you created a new file name, this creates the file on disk.
- If you're replacing a resource from an existing file, this opens a dialog box to confirm the replacement:
    - **Yes** replaces the existing file.
    - **No** returns you to the dialog box so you can specify another file name.

Caution! If you click Yes, the new file is saved on disk, replacing the existing file. Don't select Yes unless you're sure you want to replace the existing file.

Once the file is saved, its name is displayed in the editor's title bar.

Save Into

You cannot use Save Into to save a resource into a new file, nor would you want or need to. Just use Save.

Save Into saves the current resource into an existing resource file (.RES), executable file (.EXE), or dynamic link library (DLL), as well as into existing .BMP (Bitmap editor), .CUR (Cursor editor), and .ICO (Icon editor) files. In each of these last three cases, Save Into writes over the existing resource stored in the file, replacing it with the current resource.

Adding a new resource to .EXE and DLL files doesn't enable the resource. You must define functionality for the resource in your source code.

.RES, .EXE, and DLL files store multiple resources. Therefore you must assign a name to each resource you might save into one of these file types. If you assign the resource a unique name, the new resource is added to the file. If you assign a name already used by another resource in the file, Save into gives you the option of either replacing the existing resource or preserving the existing resource by assigning a different name to the current resource you are trying to save.

When you click Save Into, a dialog opens so you can name the file:

- If the file you want to save into isn't in the current directory, change the directory.
- Select the file you want. This automatically copies the file's name into the Filename field, and also selects the corresponding File Type radio button.
- Click the dialog's OK button. This adds the current resource to the selected file.

New Header

In the Accelerator, Dialog box, Menu, or String editor, the New Header command creates a new header file so you can define new constants in the header file and correlate them with the resources you define in the editor.

New Header adds a Symbol column to the editor's table, and also opens a special header-file editor. For more information on header files and instructions on editing symbol values, see Chapter 9.

Open Header

In the Accelerator, Dialog box, Menu, or String editor, Open Header opens an existing header file so you can correlate constants from the header file with the resources you define in the editor. Open Header adds a Symbol column to the editor's table, and also opens a special header-file editor. For more information on header files and instructions on editing symbol values, see Chapter 9. See also the section on header files in the Accelerator, Dialog box, Menu, and String editor chapters.

Resource Attributes

Resource Attributes opens the Resource Attributes dialog box, which lets you define attributes for the current resource.

1. Select the attributes you want to apply to the current resource; attributes that don't apply to a particular editor are disabled in the dialog box.
2. The defaults are Load On Call, Moveable, and Discardable for all resource types except bitmaps. For bitmaps, the defaults are Load On Call and Moveable. You can select the defaults for

the type of resource you're editing by simply clicking the Default button rather selecting each attribute separately.

- **Name** is the name of the resource. The name can be a number, or an alphanumeric name. If you change the name in this field, it effectively works like Save As: it saves the resource under a new name, without affecting the original resource.
- **Discardable** determines that Windows has to keep the resource in memory, even when it isn't being used. When Windows needs more memory, it discards resources defined as discardable until it has the memory it needs. If a resource isn't defined as discardable, Windows won't discard this resource when it needs memory. Typically, icons, cursors, and strings are defined as discardable because they're read-only and Windows can safely discard them. Discardable resources must also be moveable.
- **Load On Call** determines that Windows loads the resource only when it's called from the application. Choose Load on Call unless your application needs the resource immediately after beginning execution.
- **Preload** determines that this resource is loaded into memory when the application is started. Choose Preload when you know your application needs the resource immediately after it begins execution.
- **Moveable** determines that Windows can move this resource to another place in memory if it needs the memory for something else. Typically, bitmaps are moveable but not discardable. This is because Windows allows bitmaps to be modified within the program, and modified resources can't be discarded.
- **Fixed** determines that the resource cannot be moved in memory but remains in a fixed place.

3. Click OK; this accepts current selections in the dialog box.

Edit menu

The basic Edit options (Cut, Copy, Paste, and Clear) are available in all the editors. Some of the editors have additional Edit options, which are also described in this section.

The Edit menu always shows the available editing operations at the time the menu is selected. So, for example, if you haven't cut or copied anything, Paste isn't available.

Undo
[Ctrl][U]

In the Bitmap, Cursor, Dialog box, and Icon editors, Undo reverses the last edit modification you made.

Cut
[Shift][Del]

In the Bitmap, Cursor, Icon, Dialog Box, and String editors, Cut removes the selected text or a selected area and stores it in the Clipboard. The cut information can be retrieved with Paste. Use Cut to move text or a portion of a graphic from one place to another, or even to another application.

The information you cut remains in the Clipboard until replaced by another Cut or Copy, or by information you clip from another application.

Cut Row
[Shift][Del]

In the Accelerator and Menu editors, Cut Row clears the current row, rather than selected text, and copies it to the Clipboard. Otherwise, it operates the same as Cut.

Copy
[Ctrl][Ins]

In the Bitmap, Cursor, Dialog Box, Icon, and String editors, Copy copies the selected text or a selected area and stores it in the Clipboard. The copied information can be retrieved with Paste. Use Copy to copy text or a portion of a graphic from one place to another, even to another application.

The copied information remains in the Clipboard until replaced by another Cut or Copy, or by information you clip from another application.

Copy Row
[Ctrl][Ins]

In the Accelerator and Menu editors, Copy Row copies the current row, rather than selected text, and stores it in the Clipboard. Otherwise, it is the same as Copy.

Paste
[Shift][Ins]

Paste copies the contents of the Clipboard (stored by Cut, Copy, or another software program) into the editor. Contents of the Clipboard aren't affected. Thus, you can paste them again to another area, or to another application.

- In the Bitmap, Cursor, and Icon editors, define the area within which you want to paste. If the area you define is smaller than the contents of the Clipboard, Paste copies whatever fits within the bounded area, measuring from the upper left corner of both the Clipboard and the selected area.

- In the Dialog Box editor, contents of the Clipboard are copied into the work area, beginning at the cursor position.
- In the String editor, contents of the Clipboard are copied into the active String Text field, replacing the selected text in the field.

Paste Row [Ctrl][Ins] In the Accelerator and Menu editors, Paste Row copies contents from the Clipboard into an entire row rather than a single field. Otherwise, it works the same as Paste.

Note Data in one table won't make sense in a table of a different resource. Thus, you can't paste information from one type of table to another. For example, you can't paste menu data into the Accelerator, or string data into the Menu.

Clear [Del] In the Bitmap, Cursor, Icon, Dialog box, and String editors, Clear removes the selected text or a selected area *without* storing it in the Clipboard. Use Clear to erase information from the editor.

Clear Row In the Accelerator and Menu editors, Clear Row removes all text from the current row *without* storing it in the Clipboard.

Clear All In the Bitmap, Cursor, and Icon editors, Clear All clears the entire work area *without* copying information to the Clipboard.

Select All In the Dialog Box editor, Select All selects all items in the dialog box. To unselect certain items, hold down *Shift* and click each item you don't want selected.

Header menu

In the Accelerator, Dialog box, Menu, and String editors, the Header menu is disabled until you open a header file by choosing New Header or Open Header from the File menu. Creating or opening a header file also opens the header file editor. The Header menu lets you hide the menu while you're working on the resource table, and show the menu again when you're ready to use it.

Hide	Hide removes the Header menu from view so it doesn't obstruct your view of the resource table. As an alternative to choosing Header | Hide, double-click the header editor's Control-box menu.
Show	Show brings the header file back into view after it's been hidden.

A P P E N D I X

A

Troubleshooting and error messages

This appendix contains questions and answers to help you solve problems that might occur during installation or use of the Resource Toolkit. If you have technical problems, check this appendix before calling Technical Support.

Error messages

When I run the Resource Toolkit, I sometimes get the message "Cannot Find WRT.IMA" but the file seems to be there. What's wrong?

The Resource Toolkit looks for the file WRT.IMA in the same directory as file WRT.EXE. It also looks for files WRT.FON and WRT.DAT. Be sure all the Resource Toolkit's files are always kept in the same directory.

When I try to run the Resource Toolkit, I get the error message "Can't Run WRT.IMA". What does this mean?

This error message means you tried to start the Resource Toolkit with the wrong file. Review the instructions in Chapter 1 for starting the Resource Toolkit.

If you double-clicked the Resource Toolkit's icon in the Program Manager and got this message, check the properties you defined for the Resource Toolkit. To do so, make the Program Manager active, then click the Resource Toolkit's icon. Choose File Properties from the Program Manager's File menu, and be sure you entered the file WRT.EXE in the Command Line field.

Memory configuration

I get the error message "Not enough memory to run the Resource Toolkit" but I seem to have plenty of memory. What should I do?

Windows may not have enough memory to start the Resource Toolkit. Try removing unnecessary mouse or network device drivers, popup programs (such as Sidekick), and so on. Then reboot your machine and try to start the Resource Toolkit again. Alternatively, you may want to change the memory allocated to the Resource Toolkit. For instructions, see below.

I get a fatal system error message indicating there's not enough memory to work in the Resource Toolkit. Can I increase the memory?

This message is displayed when the Resource Toolkit's memory is exhausted. Normally, this should happen only when you open too many editors at once. To avoid the problem, close an editor when you're done working with it.

Controlling memory allocation

The Resource Toolkit can run on computers with as little as 1M of memory. However, many systems have much more memory, in some cases as much as 16M. This section explains how to configure the Resource Toolkit's memory usage for your system.

Static and dynamic memory settings

The Resource Toolkit allocates two different kinds of memory from the memory that Windows provides to applications. *Static* memory is allocated for the Resource Toolkit's compiled code. *Dynamic* memory is allocated for opening files and editing data. You have direct control over how much memory is allocated in each instance.

By default, the Resource Toolkit allocates 40K of memory for dynamic memory and 60K for static memory. These settings should allow the Resource Toolkit to run on any system that has at least 1M of memory. If you get a system error box that tells you there isn't enough memory to run the Resource Toolkit, see the section (on page 112) on the WIN.INI file to control memory allocation.

The Resource Toolkit is shipped in *static swapping mode*. This allows it to run on systems with limited memory by keeping compiled code on the disk until needed. Although the Resource Toolkit's compiled code takes about 350K of static memory, static swapping lets it run with a static memory allocation of as little as 60K.

Checking free memory

When the Resource Toolkit is running, you can check to see how much memory is still available to Windows and the Resource Toolkit. To check the memory, choose About WRT from the Control menu. This opens a dialog box that shows memory available to the Resource Toolkit, and also memory available to Windows. You can change the memory allocation for the next Resource Toolkit session by ending the current session and modifying the WIN.INI file, as discussed on page 112.

Windows and memory

Windows runs in one of three modes: standard, 386 enhanced, or real. The Resource Toolkit runs in standard or 386 enhanced mode. You can check to see which mode Windows is using by making the Program Manager active and choosing About Program Manager in the Help menu.

In either standard or 386 enhanced mode (both are protected mode), Windows can provide a considerable amount of memory to the Resource Toolkit and other applications. Generally, increasing static memory to 175K or higher minimizes the amount of static swapping necessary, increasing system performance. Increasing dynamic memory to 100K or higher gives the Resource Toolkit plenty of room for opening large .EXE or .RES files.

Windows memory should be at least 20K; Resource Toolkit memory should stay above 5K to 10K.

The optimum values for your particular system largely depend on how many memory resident programs are loaded into memory and what kind of memory is available. With any computer system, you should have at least 20K of Windows memory after the Resource Toolkit is started; otherwise Windows performance will worsen. Never let it fall below 10K. Normally, you'll also want at least 20K of dynamic memory available and 15K of static memory. Check free memory (as described previously), and use these values as a guide when changing the static and dynamic settings.

To increase Windows memory, you can exit applications you aren't currently using. You can also reduce the amount of memory Windows allocates to the Resource Toolkit, as discussed in the following section.

Using WIN.INI to control memory allocation

You can specify the values for static and dynamic memory in the Windows file, WIN.INI. This file contains Windows initialization data, including information about the Windows applications you will run. You can control static and dynamic memory allocation by adding a [WRT] section to the WIN.INI file. If you don't create a [WRT] section, the Resource Toolkit uses the default static and dynamic memory settings.

To make changes to WIN.INI, edit the file in a text editor and restart Windows. To specify how much memory the Resource Toolkit allocates, modify the [WRT] section using the key words **static** and **dynamic**. Doing this will *override* the values in file WRT.IMA.

The following [WRT] section confirms the default settings for WRT.IMA:

```
[WRT]
static=60
dynamic=40
```

These settings ensure that the Resource Toolkit can run on any computer. You might want to change the [WRT] section of WIN.INI to look like this:

```
[WRT]
static=150
dynamic=100
```

On many computers, you can further improve performance by setting the static memory to 175K or higher. If the memory settings are too high or too low to run the Resource Toolkit, you'll have to change the settings in the WIN.INI file to start the Resource Toolkit. The highest value you would want to set for static memory is 350K, since that would load all the compiled code into memory. Using a higher setting would be a waste of memory. The highest value allowed for dynamic memory in protected mode is 959K.

Note The Resource Toolkit has an automatic memory management system. Its memory manager requires one byte of memory for every byte of dynamic memory allocated to the Resource Toolkit.

Thus, increasing the dynamic setting by 10K effectively allocates 20K of Windows memory to the Resource Toolkit. Reducing the dynamic setting by 10K effectively returns 20K to Windows. When not working with large EXE or RES files, it's best to use the Resource Toolkit's default dynamic setting. Generally, the times it's worth increasing the dynamic setting is when you know you'll be working with one or more large files, or when you get a system error box that tells you there isn't enough memory to run the Resource Toolkit. Otherwise you might be wasting memory.

Disk space

Because the Resource Toolkit swaps compiled code to disk, it's important that you keep enough disk space free while editing in the Resource Toolkit. If you don't have enough, you won't be able to save resources. In this case, you will have to free some disk space and rebuild a resource. Generally, you should keep at least 3M free.

Palette appearance

When I run the Bitmap, Cursor, Icon, or Dialog Box editors, the palettes don't appear properly. In their place are letters. What's wrong?

This can occur if the font file WRT.FON isn't in the same directory as files WRT.EXE and WRT.IMA. Be sure all the Resource Toolkit's files are together in the same directory. If you can't find one or more of them, re-install the Resource Toolkit.

If, in fact, all the files are in the same directory, the problem might be caused because Windows is low in memory. In this case, read "Memory configuration" starting on page 110 for information on solving memory problems.

Switching editors

The Resource Toolkit pauses when I switch between editors. Is there a way to avoid these delays?

Under tight memory configurations, the Resource Toolkit automatically swaps unused editor code to a temporary disk file. When you switch editors, the Resource Toolkit swaps code from disk back into memory, and this results in a delay. To improve performance, increase the memory Windows allocates to the Resource Toolkit. For information on increasing memory, see the instructions on page 110.

You can also use a disk cache program to minimize these delays. To speed up the performance of most Windows programs, Windows includes the disk cache program SMARTDRV.SYS. Refer to your Windows documentation for information on configuring this program.

Recovering disk space

If for any reason your editing session in the Resource Toolkit ends unexpectedly, you should execute the following cleanup procedure to be sure no space on your hard disk is being taken up:

1. From the DOS prompt, type

   ```
   chkdsk /f
   ```

   and press *Enter*. This lets the CHKDSK program identify lost clusters on your disk.
2. If CHKDSK finds any lost clusters, it asks if you want to put them in a file. For example,

   ```
   10 lost clusters found in 1 chains.
   Convert lost clusters to files (Y/N):
   ```

   Type `N` to answer No to the prompt.
3. Delete all temporary files that remain on disk. These files all have a tilde (~) as the first character of the file name, and all use file extension .TMP. The DOS command

   ```
   del *.tmp
   ```

   deletes all the Resource Toolkit's temporary files from the directory.

Note The Resource Toolkit creates two types of temporary files: one type contains resources, the other type contains program code. The temporary files containing resources are always written to the directory where file WRT.EXE resides. The temporary files containing program code are written to the directory Windows uses to store temporary files. This defaults to the root directory, but the default can be overridden with the SET TEMP command in the AUTOEXEC.BAT file. If SET TEMP in AUTOEXEC.BAT doesn't name the directory where file WRT.EXE resides, temporary files containing program code are written to a different directory from those containing resources, and you have to delete the temporary files from both directories.

I N D E X

2.0

WHITEWATER
RESOURCE
TOOLKIT

BORLAND® C++

BORLAND

CORPORATE HEADQUARTERS: 1800 GREEN HILLS ROAD, P.O. BOX 660001, SCOTTS VALLEY, CA 95067-0001, (408) 438-5300.
OFFICES IN: AUSTRALIA, DENMARK, FRANCE, GERMANY, ITALY, JAPAN, SWEDEN AND THE UNITED KINGDOM ■ PART # 14MN-TCP05 ■ BOR 1969